CA-ADS®

User Guide

15.0

Computer Associates

First Edition, December 2000

Contents

How to Use This Manual

What this manual contains

- **Part I: Designing the Application** talks about how you design an application for development using CA-ADS tools and introduces the sample Department application that illustrates concepts in this manual.
- **Part II: Developing the Prototype** shows how you use CA-ADS development tools to define an executable application prototype based on design specifications.
- **Part III: Enhancing the Application Definition** shows how you develop a production application by adding process logical and work records to dialogs in the prototype application.

Who should use this manual

- New application developers who need to learn how to use CA-ADS
- Experienced application developers who are unfamiliar with CA-ADS
- Any application developers who want to look up basic steps for defining application components

How to proceed

- Go through the chapters in order when first learning how to use CA-ADS.
- Design the sample Department application.
- Build the sample Department application as you go through the manual.
- Look at supplemental information whenever necessary:
  - Appendix A: Sample Application Components lists components defined for the sample Department application, providing cross-reference information for each component.
  - Appendix B: Development Tools in the CA-ADS Environment summarizes how to use the CA-ADS application compiler (ADSA), dialog compiler (ADSC), online mapping facility (MAPC), and the Integrated Data Dictionary (IDD) menu facility.
  - Appendix C: Layout of the DEPARTMENT Record shows how the DEPARTMENT database record is defined at installation time in the demonstration database.

The sample Department application uses navigational data access. If your site primarily uses nonnavigational data access (SQL or Logical Record Facility), you may want information on equivalent nonnavigational commands. For this information, you can look at the *CA-ADS Reference* or *CA-IDMS SQL Reference* while building the Department application.

What you need to build the sample Department application

- Access to the demonstration database provided for installation at each site.
- Access to CA-ADS application development facilities, including authority to use ADSA, ADSC, MAPC, and the IDD menu facility and authority to define application components in the appropriate data dictionary.

Related documentation

When you're first getting started, you will probably use:

- *CA-ADS Quick Reference*
- *CA-IDMS Command Facility*
- *CA-IDMS Transfer Control Facility*
- *CA-IDMS Mapping Quick Reference Summary*

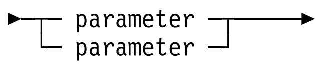
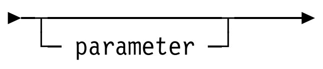
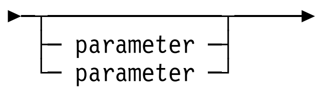
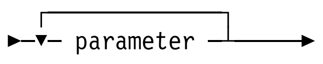
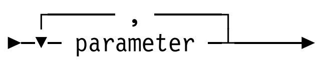
When you're more familiar with CA-ADS, you can also use:

- *CA-ADS Reference*
- *CA-IDMS Messages and Codes*
- *IDD DDDL Reference*
- *CA-IDMS Mapping Facility*
- *CA-ADS Application Design Guide*
- *CA-IDMS Performance Monitor User Guide*

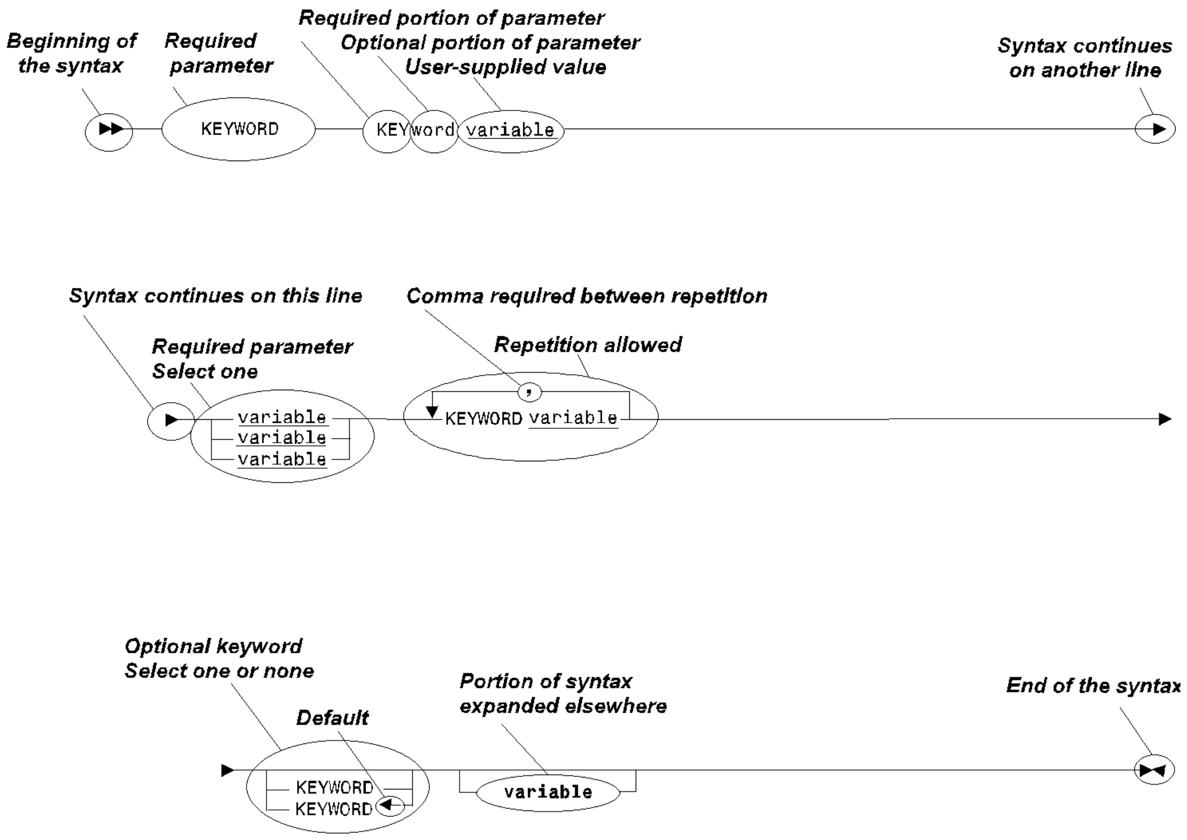
Understanding syntax diagrams

Look at the list of notation conventions below to see how syntax is presented in this manual. The example following the list shows how the conventions are used.

`UPPERCASE`
`OR`
`SPECIAL CHARACTERS`
	Represents a required keyword, partial keyword, character, or symbol that must be entered completely as shown.

`lowercase`	Represents an optional keyword or partial keyword that, if used, must be entered completely as shown.
<u>`underlined lowercase`</u>	Represents a value that you supply.
←	Points to the default in a list of choices.
`lowercase bold`	Represents a portion of the syntax shown in greater detail at the end of the syntax or elsewhere in the document.
►►────────────────────	Shows the beginning of a complete piece of syntax.
────────────────────►◄	Shows the end of a complete piece of syntax.
────────────────────►	Shows that the syntax continues on the next line.
►────────────────────	Shows that the syntax continues on this line.
────────────────────►─	Shows that the parameter continues on the next line.
─►───────────────────	Shows that a parameter continues on this line.
►── parameter ────────►	Shows a required parameter.
	Shows a choice of required parameters. You must select one.
	Shows an optional parameter.
	Shows a choice of optional parameters. Select one or none.
	Shows that you can repeat the parameter or specify more than one parameter.
	Shows that you must enter a comma between repetitions of the parameter.

Sample syntax diagram


Beginning of the syntax
Required parameter
Required portion of parameter
Optional portion of parameter
User-supplied value
Syntax continues on another line
KEYWORD
KEYword
variable
Syntax continues on this line
Comma required between repetition
Required parameter Select one
Repetition allowed
variable
variable
variable
,
KEYWORD variable
Optional keyword Select one or none
Default
Portion of syntax expanded elsewhere
End of the syntax
KEYWORD
KEYWORD
variable

Chapter 1. Building a Prototype

1.1 Overview

The development of a prototype can be approached in a variety of ways, depending upon the needs of the design team. The procedures suggested in this manual are based on a three-stage approach:

1. The initial stage performs rudimentary navigation of the application
2. The second stage begins to perform data retrieval and update
3. The final stage incorporates refinements that reflect the more complex requirements of an application running in a production environment

Each stage of the prototype is discussed below.

1.2 Stage I: Building the basic prototype

First stage: You can develop the first stage of the prototype quickly and easily because only skeletal maps and dialogs are needed for execution by the CA-ADS runtime system. Typically, you compile maps with just enough information to identify their use in the application process, and one dialog is compiled for each map. The dialogs do not need a premap process or a response process. With a minimum of time and effort, you can see how the application is going to work even before data processing takes place.

Load modules needed: To build an executable prototype, you need to provide load modules for the runtime system by:

- Compiling the application — The application and its components (the functions and responses) are defined and compiled with ADSA.
- Compiling the maps — Each map is formatted, defined, and compiled with MAPC.
- Compiling the dialogs — Each dialog is identified, associated with the appropriate map, and compiled with ADSC.

The prototype can be executed when the application, map, and dialog load modules are available for use by the CA-ADS runtime system. At this point, you have a meaningful version of the prototype that can be presented for user review and modification.

Each of the activities for building the basic prototype is discussed separately below, followed by user review considerations.

1.2.1 Compiling the application (ADSA)

The amount of detail you provide for a prototype can be as extensive as you, but the basic prototype does not have to be elaborate.

Steps in compiling an application: After signing on to ADSA, you can compile an application as follows:

1. **Identify the application** — The name of the application and related information are supplied on the Main Menu
2. **Name the task code** — The task code that designates an entry point into the application Task Codes screen. If there are multiple entry points, each task code must be defined individually.
3. **Define the responses** — The responses that initiate the functions of an application are defined on the Response/Function List and the Response Definition screens. Each response is defined on a separate screen.
4. **Define the functions** — The functions that are initiated by the responses are defined on the Response/Function List and the Function Definition screens. Each function is defined on a separate screen.

Note: Every function defined as a dialog function on the Function Definition screen in ADSA must be defined to ADSC as a dialog.

If the function is a menu, the menu type should be specified on the Function Definition (Menu) screen.

5. **Compile the application** — The application is compiled by selecting the compile option from the Main Menu

Application Definition Block and Task Activity Table: When the above-named activities are completed successfully, ADSA defines an Application Definition Block (ADB) for the application and updates the Task Activity Table (TAT). Both the ADB and the TAT are stored as load modules in the dictionary and are used by the CA-ADS runtime system when the application is executed.

1.2.2 Compiling the maps (MAPC)

Steps in compiling a map: Maps that are compiled for the first stage of the prototype usually contain all literal fields. You sign on to MAPC and take the following steps to produce the prototype screens:

1. **Identify the map** — The map name and related information are supplied on the Main Menu
2. **Format the screen** — The map design can be painted automatically using the autopaint facility of MAPC, or can be explicitly laid out using the Layout screen. The extent of the map design is left up to you. Some developers indicate the purpose of the screen with a one-line caption (for example, SCREEN FOR UPDATING EMPLOYEE RECORDS). Other developers prefer to format a screen that more closely resembles the final application version, but with literal values (such as hyphens or underscores) assigned to the variable data fields.
3. **Edit the map fields** — Each field on the map can be edited using the Field Definition screen. If all fields have been defined as literals, you can accept the default attributes by pressing [Enter] and proceed to the next editing screen; MAPC requires editing for each field on the map.

   You can take the defaults in many areas, but you must specify element names for fields. Whenever you request COMPILE, if adequate information is available, defaults are taken and the map is compiled.
4. **Compile the map** — You compile the map by selecting the compile option on the Main Menu.

A map load module is stored in the DDLDCLOD area of the data dictionary when the map has been compiled successfully.

1.2.3 Compiling the dialogs (ADSC)

Using ADSC: One dialog needs to be compiled for each map used by the prototype. To compile a prototype dialog, sign on to ADSC and specify the dialog name on the Main Menu:

In addition, specify the subschema (if there is one) and the map associated with the dialog on the Database Specifications and Map Specifications screens.

Because these are map-only dialogs, there is no need to use any other ADSC screens for the first stage of the prototype. Compile the dialog by selecting the compile function from the Main Menu. Each dialog is defined on a separate screen.

Considerations: The following considerations should be noted when compiling a dialog:

- If a dialog is defined as a function on the Response/Function List screen in ADSA, it must be defined on the Main Menu screen in ADSC (using the same dialog name).
- If a dialog is associated with a task code, it must be defined as a mainline dialog.
- The associated map must be compiled before the dialog can be compiled.

Fixed Dialog Block: ADSC defines a Fixed Dialog Block (FDB) for every dialog that is compiled successfully. The FDB is stored as a load module in the dictionary and is used by the CA-ADS runtime system when the application is executed.

1.2.4 User review

With the creation of the dialog load module, the basic prototype is ready to be presented to the user for online review. Modifications should be made to the existing prototype, the necessary load modules recompiled, and the prototype resubmitted for review until the users are satisfied.

1.3 Stage II: Adding process logic and data retrieval

Second stage: The prototype becomes more functional in the second stage. You can add activities such as the following to the prototype:

- Global records (ADSA)
- Security restrictions such as signon menus (ADSA)
- Display capabilities (MAPC and IDD)
- Premap and response process logic (ADSC and IDD)

The ADSA, MAPC, ADSC, and IDD activities used for these enhancements are described separately below.

1.3.1 ADSA enhancements

You can add the following ADSA features to the prototype at this point:

- Global records (that is, records that are available for use by all dialogs in the application) can be defined on the Global Records screen.

  Note that the ADSO-APPLICATION-GLOBAL-RECORD appears on this screen and is always included automatically as a global record.
- User-program records (that is, records that are to be passed to a user program) can be defined on the Function Definition (Program) screen, if they are needed.
- Valid responses listed for a function can be resequenced or their display can be suppressed on the second screen of the Function Definition (Menu) screen.
- Signon menu functions can be specified on the second page of the General Options screen. When security is designated as REQUIRED or OPTIONAL on this screen, the following steps should be taken:
  - Name a function as the application's signon function, using the second page of the General Options screen
  - Specify the above-named function as a menu function, using the Function Definition screen.
  - Specify that the menu is a signon menu, using the Menu Specification screen.
  - Specify the SIGNON system function as the function initiated by a response, using the Response/Function List screen.
  - Specify the response that initiates the SIGNON system function as a valid response for the named menu function, using the Function Definition (Menu) screen.

When these changes have been made, recompile the application.

1.3.2 Populating the data dictionary (IDD)

The data dictionary must contain the following components if they are to be used by the application:

- Dialog premap and response processes — Premap and response processes must be stored as process modules in the data dictionary. If premap or response processes are associated with a dialog, process modules must be defined in the data dictionary before the dialog can be compiled.

  Modules are added to the dictionary with the MODULE statement specifying LANGUAGE IS PROCESS.

- Map records and dialog work records — All work records used by a dialog and all records associated with maps must be defined in the data dictionary before the dialogs and maps can be compiled. Similarly, an application cannot be compiled unless all global records associated with the application are defined in the data dictionary. Records are added with the RECORD statement.

- Edit and code tables — All stand-alone edit and code tables associated with map records must be defined in the data dictionary before the map is compiled. Edit and code tables are added with the TABLE statement.

For complete details on adding process modules, records, and tables to the data dictionary, refer to the *IDD DDDL Reference*.

1.3.3 MAPC enhancements

Variable map fields that were specified as literals for the first stage of the prototype should be redefined as data fields and edited accordingly.

When the appropriate enhancements have been made, the map should be recompiled.

1.3.4 ADSC enhancements

You now use ADSC, recompiling the dialog to include the premap and response processes, as well as the changes made to the map associated with this dialog. After signing on to ADSC and naming the appropriate dialog on the Main Menu screen, you can add the following features:

- Work records — Supply the names of all work records associated with the dialog, using the Records and Tables screen.

  If the dialog is using subschema records, they must belong to the same subschema as the dialog. Any dialog using the ADSO-APPLICATION-GLOBAL-RECORD, must list this record on the Records and Tables screen.

- Premap process — Supply the name of the premap process associated with the dialog, using the Process Modules screen.

- Response Process — Supply the name of the response process associated with the dialog and a control key and/or response field value unique to that response process, using the Process Modules screen.

Recompile the dialog after making the appropriate enhancements.

1.4 Stage III: Refining the maps and processes

Third stage: The final stage of development can focus on refinement of the map design and the map field attributes. Some of the following additions can be made:

- Incorporate additional fields in the maps
- Add or change map field attributes
- Specify automatic editing on selected map fields
- Provide informational messages
- Add error messages

Chapter 2. Designing Maps

2.1 Overview

Maps displayed during the execution of the application interface directly with the terminal operator and, therefore, can influence the success of an application. Consequently, you must consider the appearance of the menu screens and the layout of the dialog maps.

Successful map design: A successful map design should exhibit the following attributes:

- Consistency — Entities (for example, fields, headings, labels, responses, messages, and control keys) should have the same meaning or effect throughout the application. The meaning or effect need not be identical for every map, but should be consistent within the broader confines of the system. In general, there are two special fields on any screen: a message field and a response code field. These areas should appear in a constant location on the screen throughout any application; for maximum effectiveness, they should remain standard for all applications at a site.
- Convenience — Features of the system should be designed to associate related entities by using similar constructs, positioning, and responses to produce similar reactions from the system. For example, assign one particular control key to initiate the update function in all the dialogs of a given application.
- Supportiveness — The reactions of the system should enable the user to handle normal contingencies conveniently. Tutorial aids should be available when needed. Displayed informational and/or error messages should be meaningful.

The remainder of this chapter discusses the following aspects of map design:

- Standards to consider when designing maps
- Mapping procedures that can be adopted by an installation
- Choices available in the design of menu maps
- Suggestions for designing dialog maps

2.2 Design standards for a dialog map

You need to consider the following standards when designing dialog maps:

- Design the map with the terminal operator in mind. For example, a very dense screen is tiring and difficult to use. In general, the screen most pleasing to the eye is about 40% full.
- The placement of fields on the screen, the use of high intensity, and the neatness of the format have a great deal of impact on the effectiveness of the system.
- When the screen is sent to the terminal, the cursor should be in the position most likely to be used for data entry. Other frequently used fields should be easily accessible with the tab and return keys.
- The sequence of fields, when tabbed, should match the most common pattern used for data entry.
- Fields requiring special attention should be highlighted and clearly visible.
- The screens should be as uncluttered as possible. The common error of using one screen format for excessive and/or dissimilar functions tends to produce cluttered or busy screens; separate screens with some common fields are more usable.
- Terminal users should be able to initiate processing by typing in the necessary data and pressing a control key. They should not be required to make decisions that could have been incorporated in program logic, nor should they be forced to use control keys or responses needlessly.

2.3 Mapping procedures

The following illustrate the mapping procedures that might be implemented by a specific site:

- Have one individual (for example, the data administrator) responsible for creating and modifying all maps.
- As much as possible, use the features of MAPC to handle editing, error handling, error messages, and modifying field attributes.
- Use a standard map template. Whenever possible, keep data fields in columns and double space rows of data.
- Use the BRIGHT attribute to contrast items on the screen that have different uses (for example, highlight required fields). Be consistent in the use of attributes.
- Use the cursor in a consistent manner. For example, either place the cursor at the first field to be used for data entry or at the field where the terminal operator is to enter the next function.
- Use the BRIGHT attribute for redisplaying data fields that are in error.

2.4 Choosing menu maps

When designing an application, you need to decide if system- or user-defined menu maps are to be used.

The **system-defined menu** provides a standard format for the information provided by the developer during the definition of the functions and responses of the application in an ADSA session. If a format other than the standard format is desired (for example, you want to redefine certain literal fields on the map or wants to supply site-specific headers), the **user-defined menu** map is used. Both types of maps are discussed separately below.

2.4.1 System-defined menu maps

Menu formats: If the menu map is to be system-defined, you have the option of using one of the following menu formats:

- Short description menu map (ADSOMUR1) — The menu screen that lists 30 valid menu responses per page; a short (12-byte) textual description is displayed for each response.
- Long description menu map (ADSOMUR2) — The menu screen that lists 15 valid menu responses per page; a long (28-byte) textual description is displayed for each response name.
- Signon menu map (ADSOMSON) — The menu screen that requires an CA-IDMS/DC or CA-IDMS/UCF validation of user ID and password before the menu request can be processed. The standard signon menu map can have 12 valid menu response names per page with 28 bytes of descriptive text displayed for each.

For detailed information about compiling these maps, refer to the *CA-ADS Reference*.

If none of the above menus meets the needs of the user, the system-defined menu map can be altered by the user or a new menu (designated as a menu/dialog function) can be formatted. Both methods of creating user-defined maps are discussed below.

2.4.2 User-defined menu maps

When user-specific modifications to the existing system-defined menu maps are necessary, you can change the menu maps by using either of the following techniques:

- Reformatting and recompiling the standard system-defined menu
- Designing a menu/dialog (that is, a menu map that is part of a menu/dialog function)

Each of these methods is discussed below.

2.4.3 Reformatting and recompiling the system-defined menu

You can reformat and recompile the existing system-designed menu map, retaining the same name. This method allows you to use the standard menu function rather than designing and using a menu/dialog function.

Steps to reformat the system menu: Take the following steps to reformat the system menu:

1. Obtain the source for the map being used (that is, ASDSOMUR1, ADSOMUR2, or ADSOMSON) from the source data sets created when the distribution tape was installed. The maps are stored as members under their own names.
2. Use the batch mapping compiler to store the source in the data dictionary.
3. Use MAPC to modify and recompile the menu map.

Considerations: When recompiling a menu map with MAPC, observe the following rules:

- ADSO-APPLICATION-MENU-RECORD is a required map record. Optionally, the menu can map to additional records, but it must always map to the ADSO-APPLICATION-MENU- RECORD.
- The menu must contain the same number of responses per page as the number of responses for the selected map (that is, 30 for ADSOMUR1, 15 for ADSOMUR2, or 12 for ADSOMSON).
- The AMR-RESPONSE field of the ADSO-APPLICATION- MENU-RECORD record is a required field. The first response name on the map must map to the first occurrence of AMR-RESPONSE. Each subsequent response name must map to the next corresponding occurrence.
- The AMR-USER-ID and AMR-PASSWORD fields of the ADSO-APPLICATION-MENU-RECORD are required on a signon menu map. The user ID data field must map to AMR-USER-ID, and the password data field must map to AMR-PASSWORD.
- All other fields on the ADSO-APPLICATION-MENU-RECORD are optional. The map data fields that are used must be associated with the appropriate fields on the record (for example, heading data must map to AMR-HEADING).

  If using the AMR-KEY field, note that this field appears as a single byte (the AID byte) in the ADSO-APPLICATION-MENU- RECORD. The AMR-KEY field is associated with a code table (ADSOAIDM) that translates the AID byte to more easily readable characters (for example, 1 translates to PF1, % translates to PA1).

Refer to the *CA-IDMS Mapping Facility* for more information on using MAPC to recompile a map.

2.4.4 Designing a menu/dialog

You can design and compile an entirely new menu with MAPC. This map must be defined as a menu/dialog function of the application.

Steps for defining a menu/dialog function: Follow these procedures when defining a menu/dialog function:

1. Design and compile the map using MAPC. Observe the following rules when compiling the map:
   - ADSO-APPLICATION-MENU-RECORD must be one of the records associated with the map.
   - The AMR-RESPONSE field is required for all menus. The number of required occurrences depends on the number of responses per page (to a maximum of 50) specified on the ADSA Menu Specification screen. The first response name on the map must map to the first occurrence of AMR-RESPONSE; each subsequent occurrence must map to the next corresponding occurrence of AMR-RESPONSE.
   - The AMR-USER-ID and AMR-PASSWORD fields are required for signon maps. The user ID data field must map to AMR-USER-ID, and the password data field must map to AMR-PASSWORD.
   - All other fields on the ADSO-APPLICATION-MENU- RECORD are optional. The map data fields used must be associated with the appropriate fields on the record (for example, heading data must map to AMR-HEADING).
2. Add the process source to the data dictionary in an IDD session. (The dialog associated with the menu does not have to include any process code, although the choice of a menu/dialog function suggests that some processing is intended.)
3. Compile the dialog in an ADSC session, associating the map and any processes with the dialog using the ADSC Map Specifications screen. Note that the dialog must be compiled to include the map before the application can be executed at runtime.
4. Define the dialog as a menu/dialog function for the application, using the ADSA Response/Function List screen.

An installation can develop standard map templates and the associated boilerplate code for site-specific menu/dialogs. When a menu is needed, programmers can obtain a copy of the template/boilerplate, fill in the appropriate fields and the edit/code tables needed for those fields, and submit it to the data administrator for approval.

2.5 Designing dialog maps

Design questions: Each dialog map is associated with its own dialog and must be designed to reflect the function of the associated dialog. The application specifications developed during the initial design stages can be used to answer design questions such as the following:

- How many of the dialogs specified for this application will require maps?
- What premap and response processes are required for each map?
- What job is performed by each process?
- Will the map be used to pass data between processes and/or between dialogs? What data will be passed?
- What database and mapping work records are associated with the map?
- What editing criteria should apply to the map fields?

Map templates: Just as site-specific standards can be established for menu/dialogs, an installation can use map templates to standardize the formatting of maps associated with dialog functions. Programmers can obtain a copy of the template; fill in the appropriate fields, indicating the corresponding map record fields; and submit this information to the data administrator. The data administrator can then add the necessary map design, map records, and edit/code tables (if any) to the data dictionary.

Chapter 3. Designing Dialogs

3.1 Overview

A dialog is a unit of work within an CA-ADS application that enables interaction with the terminal operator. Because dialogs are the basic building blocks of an CA-ADS application, it is important that they be well-designed. This chapter discusses characteristics and design features of dialogs that merit the attention of application developers.

3.2 Dialog characteristics

The characteristics of a dialog determine its role within the application. Each dialog has an implicit level and status, and can pass and receive control of the processing. The significance of the dialog level and status and the manner in which control is passed are discussed below.

3.2.1 Dialog level

The level of a dialog refers to its position within the application structure. You can pass processing control to a dialog at the next lower level, the same level, the next higher level, or the top level of the application structure.

Note: The meaning of TOP changes whenever a LINK command is executed. The dialog issuing LINK becomes the current TOP.

Impact of the dialog level: At runtime, the dialog level affects the following aspects of an application:

- Availability of data — When combined with the manner in which processing control is received, the level of a dialog governs the data passed in the record buffer blocks and the currencies that are established, saved, stored, or released.
- Use of system resources — The runtime system maintains record buffer blocks, database currency blocks, and variable dialog blocks for dialogs at each level. There is a direct correlation between the number of dialog levels in an application and the size of the storage pool that is needed.
- Performance — The number of dialog levels can affect the performance of an application. For example, performance times are affected if a frequently accessed dialog is located three or four levels down in an application structure.

An application can be composed of any number of dialog levels, but the most efficient application uses many levels only when absolutely necessary.

Mainline dialog: The top-level dialog must be a **mainline** dialog and must be defined as such by the application developer. A mainline dialog is the entry point to the application. An application can have more than one mainline dialog; entry points can also be established at a lower level in the application structure. In addition to defining a task code for the top-level dialog, the developer can identify an alternative entry point by using the Task Codes screen to associate a task code with a lower-level function.

3.2.2 Dialog status

Operative and nonoperative status : A dialog can have an **operative** or a **non-operative** status within the application thread. A dialog becomes operative when it receives control and begins executing; at a given level, only one dialog can be operative at a time. When control passes to a dialog at another level, the issuing dialog can remain operative or can become nonoperative, depending upon the level of the next dialog. For example, when control is passed with the LINK command, the issuing dialog remains operative; when control is passed with the TRANSFER command, the issuing dialog becomes nonoperative.

As long as a dialog is operative, all data that it has acquired is retained. When a dialog becomes nonoperative, its data is released. See "Database currencies" later in this chapter for a summary of the way in which a dialog's status is affected by the successful execution of a control command.

Application thread: Within the application structure, only one dialog executes at a time. The sequence of dialog execution within an application structure is called the **application thread**. The response of the terminal operator determines the dialogs that constitute a given application thread. The following diagram shows an application structure and one application thread.

One dialog can exist in several places within the application structure and be part of the same or different application threads. A dialog can execute more than once within the application thread whether or not it remains operative.

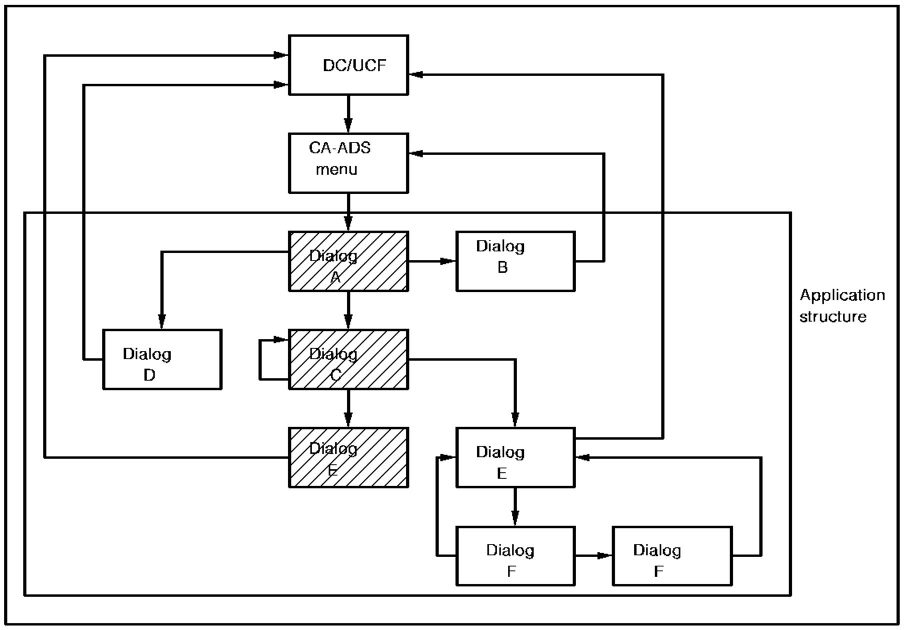
In the diagram below, the shaded boxes represent an application thread that includes dialog A, dialog C, and dialog D.

3.2.3 Dialog control

A dialog passes control to another dialog based on the execution of a control command and/or the terminal operator's selection of processing. The dialog that receives control can be a different dialog, a copy of the executing dialog, or all or part of the executing dialog itself.

Operations performed by control commands: You can use specific control commands to perform the following operations:

- Pass processing control from one dialog to another dialog or to a user program
- Display a dialog's map
- Terminate an existing dialog or application
- Exit CA-ADS
- Pass processing control to specified points within a dialog and reinitialize the record buffers associated with a dialog



Most of the control commands used are available to all applications. When designing dialogs that will become part of an ADSA application, you can also use the EXECUTE NEXT FUNCTION command.

3.3 Design considerations

You need to keep the following CA-IDMS/DB, CA-IDMS/DC and CA-IDMS/UCF (DC/UCF), and CA-ADS features in mind when designing the dialogs:

- Database currencies
- Record buffer management
- Working storage areas
- Extended run units
- Longterm locks
- Global records

Each of these issues is presented below.

3.3.1 Database currencies

In CA-ADS, currency is maintained automatically for the user. To facilitate this feature, a currency control block is created that maintains currency information. At runtime, a currency block is created for each dialog in the application structure that performs database requests.

Passing currencies: Database currencies are passed from one dialog to another dialog at a lower level, enabling dialogs to continue database processing from an established position in the database. Currencies are cumulative. The currencies established by each dialog are passed to lower-level dialogs, which, in turn, establish their own currencies; the cumulative currencies are passed to the next lower-level dialog.

Currencies are established, saved, restored, and released as follows:

- Established — Currency is established with the dialog's first functional database call. Established currencies are updated when database commands (for example, FIND, OBTAIN, ERASE) are encountered during the transaction. Currency is nulled when a dialog receives control with a RETURN or TRANSFER command.
- Saved — When a LINK, DISPLAY, or INVOKE command is issued, the database currencies established with the last database command in the dialog are saved. Saved currencies are available to lower-level dialogs and are restored to the issuing dialog if processing control returns.
- Restored — Saved currencies are restored when CA-ADS opens a transaction in the dialog receiving control (that is, saved currencies are restored just prior to the first database call).
- Released — When a LEAVE, RETURN, or TRANSFER command is issued, all database currencies at the same and lower levels are released. The dialog receiving control must establish its own currencies or use the currencies passed to it from another higher-level dialog.

Note that currencies, as described in this section, pertain only to DML run units. SQL run units are not managed by the CA-ADS runtime system. Please refer to the SQL programming guide for more information on SQL programming techniques.

The following table shows the ways in which the passing and receiving of control affects the contents of the currency block.

Command	**New level established**	**Status of issuing dialog**	**Data available to receiving dialog**	**Currency action**	
				Issuing dialog	**Receiving dialog**
DISPLAY	No	Operative	All data	Saved	N/A
INVOKE	Yes	Operative	All data	Saved	Restored
LEAVE	No	Non-operative	No data	Released	Must establish
LINK					
DIALOG	Yes	Operative	All data	Saved	Restored
PROGRAM	No	Operative	All, some, or none, (depending on command specification)	Saved	Must establish
RETURN	No	Non-operative (any operative dialogs between the issuing dialog and the receiving dialog also become non-operative)	Data previously acquired by the receiving dialog	Released (currencies for any dialogs between the issuing dialog and the receiving dialog are also released)	Restored
TRANSFER	No	Non-operative	All data except that acquired by the issuing dialog	Released	Can use currencies previously established by higher-level dialogs

3.3.2 Record buffer management

Record buffer management is affected by whether the record used by the dialog is a database, work, or map record; a logical record; or a record that has been assigned the NEW COPY attribute. The manner in which the CA-ADS system allocates space for these records in the Record Buffer Block (RBB) is discussed below.

3.3.3 Database, work, and map records

At the beginning of each application thread, the CA-ADS runtime system allocates a primary Record Buffer Block (RBB) and initializes a buffer in the RBB for each record associated with the top-level dialog.

Considerations: All lower-level dialogs can access records in any of the existing buffers, unless one of the following conditions is true:

- The dialog that receives control accesses a database record or a work record that has been assigned the NEW COPY attribute during dialog compilation.
- The dialog that receives control accesses a database record, work record, or logical record not used by a higher-level dialog.
- The dialog that receives control accesses a database record that uses a subschema not used by a higher-level dialog.

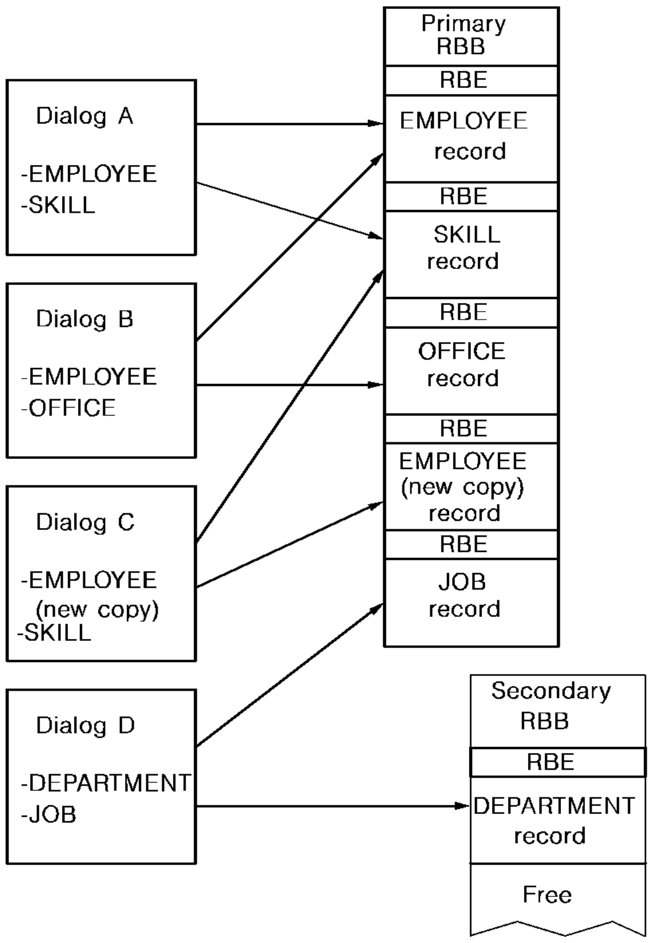
If one or more of these conditions exist, CA-ADS allocates and initializes an additional buffer for the record.

Additional buffers: Additional buffers are also allocated and initialized when one of the following situations exists:

- The record is assigned the WORK RECORD attribute during dialog compilation.
- The record is associated with the map used by the dialog.
- The record is named explicitly in a database command.

The following diagram shows the sequence in which CA-ADS initializes record buffers as a series of dialogs receives control.

When dialog A begins executing, CA-ADS allocates buffers for the EMPLOYEE and SKILL record types. Dialog B uses the previously allocated EMPLOYEE record buffer, but requires a new buffer for the OFFICE record. Dialog C requests and receives a new copy of the EMPLOYEE record buffer, but uses the previously allocated SKILL record buffer. Dialog D requires new buffers for both the DEPARTMENT and JOB records. CA-ADS allocates a secondary RBB to accommodate the DEPARTMENT record, but uses the remaining space in the primary RBB for the JOB record.



3.3.4 Logical records

Considerations: When an application thread contains dialogs that use a combination of database records and logical records, special considerations apply with respect to record buffer management. For each database record component of a logical record, CA-ADS initializes individual, contiguous record buffers. The logical record components are placed in the buffer in the order named in the logical record definition.

For example, consider the EMP-JOB-LR logical record, which consists of four database records: EMPLOYEE, DEPARTMENT, JOB, and OFFICE records. If dialog B accesses EMP-JOB-LR, CA-ADS initializes new record buffers for each of the four records listed above (in that order) *regardless* of whether buffers for one or more of the records were initialized when dialog A, a higher-level dialog, began executing. Therefore, dialog B (and lower-level dialogs accessing the same logical record) does not have access to data established in the record buffer by dialog A. However, dialogs at levels lower than dialog B will use the buffers established by dialog A if those dialogs use the same database records as dialog A.

When using both database records and logical records, the first dialog of the application thread should include an INITIALIZE command for the logical record. This action associates the logical record with the top-level dialog and ensures that the buffer for the entire logical record will be allocated and available to all lower-level dialogs. Lower-level dialogs will use the component record buffers established at the highest level unless the logical record itself is referenced.

3.3.5 NEW COPY records

Records can be assigned the NEW COPY attribute on the Records and Tables screen during the definition and compilation of a dialog. The NEW COPY designation signifies that the record in question is to receive newly initialized record buffers when the dialog is executed.

The NEW COPY attribute is used when the programmer wants to obtain another occurrence of a record type without overwriting the data that is in the current buffer. To have the use of a second, temporary buffer for the same record type, the programmer links to a lower-level dialog that has specified NEW COPY for that record. An occurrence of the record type is brought into the new buffer and processed as directed. When control returns to the calling dialog, the record buffer at the upper level contains the same data as before; the data in the lower-level record buffer is no longer available.

Dialogs at a level lower than the dialog with a NEW COPY record will not use the NEW COPY buffer, but will use the first buffer allocated for the record.

3.3.6 Working storage areas

DC/UCF queue and scratch areas can be used by the CA-ADS dialogs as working storage areas. The methods by which dialogs can store and use records in the queue and scratch areas are presented below.

3.3.7 Queue records

Queue records can be used as work records that are shared by tasks on all DC/UCF terminals. Entries are directed to a queue with database commands embedded in the dialogs or batch programs. Queues can transfer data across the entire DC/UCF system and are maintained across system shutdowns and crashes. Currencies and locks are not passed between tasks.

Characteristics: Queue records have the following characteristics:

- A queue header record is allocated either at system compilation or by an application dialog.
- Queue records participate in a set in the data dictionary; this set is commonly referred to as a queue.
- Queue records are locked by each task; no other task can use them until the locks are released.

Queues created at system compilation with the system QUEUE statement can be accessed by an CA-ADS application. Additionally, an application can create its own queues by requesting storage space with a GET QUEUE statement in the dialog process code.

Functions: An application can use queue records to accomplish the following functions:

- Automatically initiate a task — DC/UCF initiates a task that processes the queue entries when the number of entries in a queue reaches a specified limit or when a specified time interval has passed. For example, an application can write records to a queue and the system will route the records to a printer when the collected records exceed the specified limit.
- Avoid prime time updating — Records that need to be updated can be collected on a queue; the queue can be accessed by a batch program at a low-use time.
- Prevent run-away tasks — A maximum limit can be established for the number of entries permitted in a queue. The UPPER LIMIT parameter of the QUEUE statement is especially useful in a test environment to prevent a looping program from filling the scratch/queue area.

For detailed descriptions of the queue management commands, refer to the *CA-ADS Reference*.

3.3.8 Scratch records

Scratch records are shared between tasks and saved across the transactions of an CA-ADS application. Used as a temporary storage area, scratch records provide a means of passing data between tasks running on the same terminal; they are not accessible to tasks that execute on other terminals and are not saved across a system shutdown or a system crash.

Characteristics: The following characteristics are associated with scratch records:

- Scratch records are stored in the data dictionary.
- Multiple scratch areas are allowed for a task and multiple records can be maintained within a scratch area.
- Currency is maintained for each area and record, and can be passed between tasks.
- The scratch area is allocated dynamically within the storage pool. When all scratch records are deleted, the area will also be deleted.

Functions: Scratch records can be used in the following ways within an application:

- To save input acquired from two or more dialogs over the course of the application.
- To allow multiple occurrences of a record to be mapped out at one time. For example, if the names, addresses, and phone numbers of all department employees need to be mapped onto the same screen in multiples of five, the following steps could be taken:

1. Walk the set of employee records, moving the required data to a work record that contains multiply-occurring fields.
2. When the work record contains the data on five employees, move the contents of the work record to the scratch area with a PUT SCRATCH command so that, in effect, a screenful of data on five employees is put on each record in the scratch file.
3. Walk the set of scratch records when the screens of information are to be displayed.

- To pass the contents of the record buffer when a dialog receives control with a TRANSFER command. Data acquired by the dialog issuing a TRANSFER command is not available to the dialog receiving control. However, the dialog receiving control could access buffer data that had been placed in a scratch record.

Refer to the *CA-ADS Reference* for detailed descriptions of the scratch management commands.

3.3.9 Extended run units

Typically, an CA-ADS transaction begins when the dialog issues a command accessing the database (for example, OBTAIN) and ends when the runtime system encounters the next control command issued by the dialog (that is, LINK, INVOKE, DISPLAY, TRANSFER, LEAVE, or RETURN). An extended transaction is a transaction that is kept open when the runtime system encounters the LINK command under the following circumstances:

- When the LINK is to the premap process of a dialog with no associated subschema
- When the LINK is to the premap process of a dialog with an associated schema and subschema identical to those of the calling dialog
- When the LINK is to a user program

Implications: Implications of the extended transaction are as follows:

- Currencies are passed to the lower-level dialog and are restored upon return to the upper-level dialog.
- Currencies are not passed to user programs; currencies are saved and restored to the upper-level dialog when control is returned.
- The lower-level dialog can perform error checking to decide whether to issue a ROLLBACK command.
- Because a FINISH is not issued, record locks held by the upper-level dialog are not released. A COMMIT can be coded in the upper-level dialog if the developer needs to release locks before linking to the lower-level dialog.
- If a COMMIT is issued prior to the LINK command and an abend occurs in the lower-level dialog, the rollback will be incomplete; the rollback will only go to the COMMIT checkpoint and not to the start of the transaction.

- If a lower-level user program opens its own transaction, a deadlock can occur. The possibility of a deadlock condition can be avoided by taking either of the following actions:
  - Issue a COMMIT prior to the LINK.
  - Pass the subschema control block to the user program and let the program use the same transaction. Issue no BINDs or FINISHes in the user program.

For more information on the extended transaction, refer to the *CA-ADS Reference*.

3.3.10 Longterm locks

The KEEP LONGTERM command sets or releases longterm record locks. Longterm locks are shared or exclusive locks that are maintained across transactions. Once the longterm locks are set, all other transactions are restricted from updating or accessing the named records until the dialog explicitly releases the locks. The following example requests the release of all longterm locks associated with the current task:

```
KEEP LONGTERM ALL RELEASE
```

Monitoring database activity: The KEEP LONGTERM command can also be used to monitor the database activity associated with a record, set, or area. When a dialog is updating records that could also be updated by another user, the following code can be included in the premap process of the named dialog:

```
KEEP LONGTERM longterm-id NOTIFY CURRENT record-name
```

This command instructs the CA-ADS runtime system to monitor the database activity associated with the current occurrence of the named record type.

The following code is included in the response process of the same dialog:

```
KEEP LONGTERM longterm-id


TEST RETURN NOTIFICATION INTO return-location-v
```

This command requests notification of any database activity against records that were specified in the KEEP LONGTERM premap process. If appropriate, the dialog can check the return value placed in the specified work record field.

For more information on the KEEP LONGTERM command, refer to the *CA-ADS Reference*.

3.3.11 Global records

Global records are records that are available to all dialogs, maps, and user programs in an application. Subschema records cannot be defined as global records.

System-defined global record: The ADSO-APPLICATION-GLOBAL-RECORD is the system-defined global record that enables communication between the application and the runtime system. To be accessed by a dialog, the ADSO-APPLICATION-GLOBAL-RECORD must either be specified as a dialog work record or be associated with the dialog's map. This record is initialized when an application is first loaded by the runtime system.

All fields in the ADSO-APPLICATION-GLOBAL-RECORD are addressable by dialogs or user programs. Selected fields from the ADSO-APPLICATION-GLOBAL-RECORD are listed below. For a complete listing of these fields, refer to the *CA-ADS Reference*.

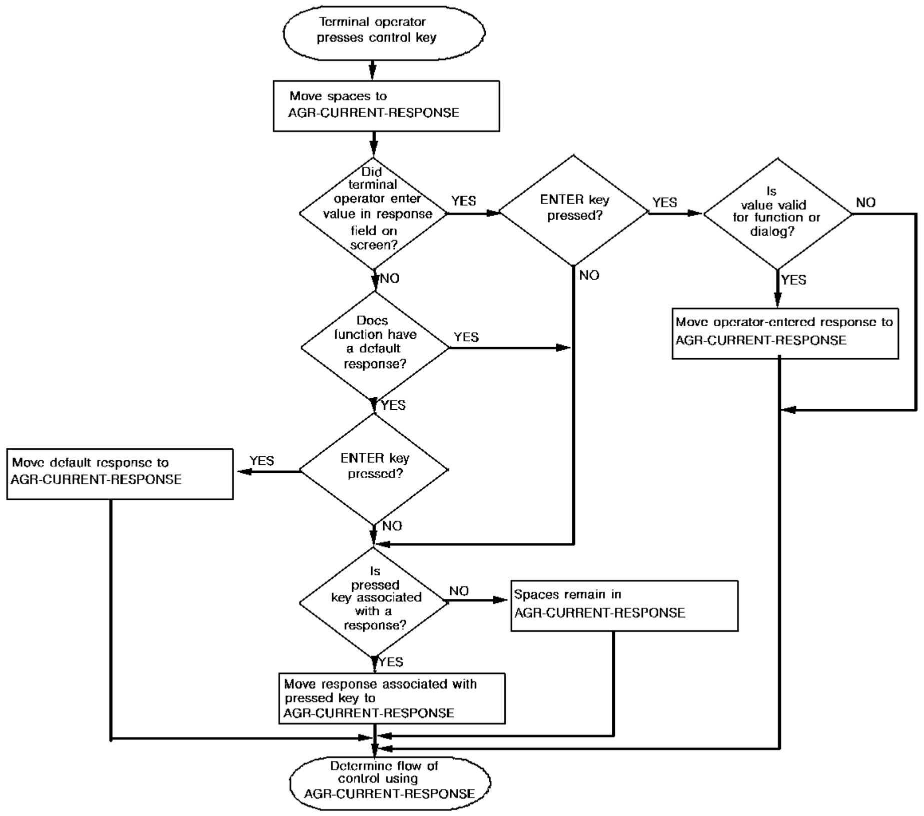
AGR-NEXT- FUNCTION: The AGR-NEXT-FUNCTION field contains the name of the next function that is to be executed. When the dialog associated with the current function ends with an EXECUTE NEXT FUNCTION command, the function named in the AGR-NEXT-FUNCTION field is executed by the runtime system. A dialog or user program can query this field to check what the next function will be. Modification of the AGR-NEXT-FUNCTION field, however, does not change the next function to be executed; a change in the next function can only be accomplished by modification of the AGR-CURRENT-RESPONSE field (see below).

AGR-DEFAULT- RESPONSE: The AGR-DEFAULT-RESPONSE field contains the default response value specified on the Function Definition screen when an application is compiled. When a value is specified and the screen includes a data field for a default response, the terminal operator can type in a new value or can space out the value that appears.

AGR-CURRENT- RESPONSE: The AGR-CURRENT-RESPONSE field contains the response specified by the terminal operator. The process code of a dialog or user program can also move values into this field, overwriting the user response. Note that, if AGR-CURRENT-RESPONSE is modified by a dialog, security is not checked for the response moving into the field, even if security is associated with this response.

When EXECUTE NEXT FUNCTION is encountered within process code, the response named in the AGR-CURRENT- RESPONSE field is executed if it is a valid response for the current function. The AGR-CURRENT-RESPONSE field determines the next function in the application thread (that is, it determines the value moved into the AGR-NEXT-FUNCTION field).

Moving a value into AGR-CURRENT- RESPONSE: The following diagram shows the manner in which the runtime system moves a value into the AGR-CURRENT-RESPONSE field. The value in AGR-CURRENT-RESPONSE depends upon whether the AGR-DEFAULT-RESPONSE field contains a value; whether the terminal operator enters a new value in the response field; or whether there is a response value associated with the control key (other than ENTER) pressed by the terminal operator. The runtime system executes the response named in the AGR-CURRENT-RESPONSE field after determining that it is a valid response for the current function.


Terminal operator presses control key
Move spaces to AGR-CURRENT-RESPONSE
Did terminal operator enter value in response field on screen?
YES
NO
ENTER key pressed?
YES
NO
Is value valid for function or dialog?
NO
YES
Move operator-entered response to AGR-CURRENT-RESPONSE
Does function have a default response?
YES
YES
ENTER key pressed?
YES
NO
Move default response to AGR-CURRENT-RESPONSE
Is pressed key associated with a response?
NO
YES
Spaces remain in AGR-CURRENT-RESPONSE
Move response associated with pressed key to AGR-CURRENT-RESPONSE
Determine flow of control using AGR-CURRENT-RESPONSE

AGR-EXIT-DIALOG: The AGR-EXIT-DIALOG field initially contains the name of the exit dialog specified on the Application Definition screen. This field can be used to link to a special routine. For example, one department of a company might want the employee name specified as John Doe, while another department wants the name specified as Doe, John. The same dialog could be used for both departments by linking to an exit dialog (that is, LINK TO AGR-EXIT-DIALOG) containing a name routine.

AGR-PRINT- DESTINATION: The AGR-PRINT-DESTINATION field initially contains the default name of the printer for the application as specified on the ADSA Application Definition screen. Dialogs and user programs can use this print destination with the WRITE PRINTER DESTINATION command.

AGR-USER-ID: The AGR-USER-ID field can be queried by dialogs and user programs.

AGR-PRINT-CLASS: The AGR-PRINT-CLASS field initially contains the default printer class for the application as specified on the ADSA Application Definition screen. The dialog can reference this field with the WRITE PRINTER CLASS command.

AGR-SIGNON- SWITCH: The AGR-SIGNON-SWITCH field can be queried to determine if there has been a valid signon.

AGR-SIGNON- REGMTS: The AGR-SIGNON-REQMTS field indicates whether signon is optional, required, or not used for the signon menu, as specified on the Security screen. This field can be referenced for additional security checking.

AGR-MAP- RESPONSE: The AGR-MAP-RESPONSE field can be used as a response field, in place of the $RESPONSE field, in any user-defined nonmenu map. The dialog can initialize this response field before mapout so that the desired default response appears on the map. For input purposes, the AGR-MAP-RESPONSE field works in the same manner as the $RESPONSE field. For information on the $RESPONSE field, refer to the *CA-IDMS Mapping Facility*.

AGR-MODE: The AGR-MODE field initially contains the value STEP or FAST as specified on the Application Definition screen. Typically, the design of a dialog map includes a field that displays the value of AGR-MODE. The terminal operator can change this field at any time.

The following examples show how the AGR-MODE field can be used, in conjunction with the EXECUTE NEXT FUNCTION command, to implement a STEP/FAST mode for an ADSA application. The logic in the first example assumes that all data field validation is handled by the automatic editing specifications in the dialog's map. The logic in the second example assumes that additional data validation is required in the response process code. In both cases, any data entered by the terminal operator is always processed. Note that the first pass flag field has no significance in FAST mode.

Example 1: This sample process code illustrates the manner in which a dialog can query the AGR-MODE field of the ADSO-APPLICATION-GLOBAL- RECORD to determine what course to follow. If the dialog is in STEP mode, the dialog redisplays the screen with a confirmation message for the terminal operator; if in FAST mode, control is passed immediately to the next function. The initial value of AGR-MODE is supplied by the runtime system; the user can alter the value of AGR-MODE at any time during application execution.

```
IF ANY OF (EMPLOYEE-NBR, SKILL-CODE, SKILL-LEVEL)
  ARE CHANGED
   DO.
     MOVE 'Y' TO FIRST-PASS-FLAG.
     MOVE EMPLOYEE-NBR TO WK-EMPNBR.
     MOVE SKILL-CODE TO WK-SKLCODE.
     MOVE SKILL-LEVEL TO WK-SKLEVEL.
     LINK TO 'CEMDUEMP'.
   END.
IF AGR-STEP-MODE
   DO.
     IF FIRST-PASS-FLAG='Y'
       DO.
         MOVE 'N' TO FIRST-PASS-FLAG.
         DISPLAY MSG TEXT IS 'EMPLOYEE UPDATED'.
       END.
     MOVE 'Y' TO FIRST-PASS-FLAG.
   END.
EXECUTE NEXT FUNCTION.
```

Example 2: The sample code shown in the following example shows the use of the AGR-MODE field when data validation needs to be handled by code in the response process. Note that the EXECUTE NEXT FUNCTION command is never encountered while uncorrected validation errors still exist.

```
IF ANY OF (EMPLOYEE-NBR, SKILL-CODE, SKILL-LEVEL)
  ARE CHANGED
   DO.
     MOVE 'Y' TO FIRST-PASS-FLAG.
     IF EMPLOYEE-NBR GE 2000 AND SKILL-CODE='A'
        DO.
          MOVE 'Y' TO ERROR-FLAG.
          DISPLAY MSG TEXT IS
             'EMPLOYEE NUMBER/SKILL CODE MISMATCH'.
        END.
     MOVE 'N' TO ERROR-FLAG.
     MOVE EMPLOYEE-NBR TO WK-EMPNBR.
     MOVE SKILL-CODE TO WK-SKLCODE.
     MOVE SKILL-LEVEL TO WK-SKLEVEL.
     LINK TO 'CEMDUEMP'.
     CALL EMPDTE25.
   END.
IF ERROR-FLAG='Y'
    DISPLAY MSG TEXT IS
       'EMPLOYEE NUMBER/SKILL CODE MISMATCH'.
CALL EMPDTE25.
!***************************************************
DEFINE EMPDTE25.
!***************************************************
IF AGR-STEP-MODE
   DO.
     IF FIRST-PASS-FLAG='Y'
       DO.
         MOVE 'N' TO FIRST-PASS-FLAG.
         DISPLAY MSG TEXT IS 'EMPLOYEE UPDATED'.
       END.
     MOVE 'Y' TO FIRST-PASS-FLAG.
   END.
EXECUTE NEXT FUNCTION.
```

The following fields from the ADSO-APPLICATION-GLOBAL- RECORD are often mapped to screens associated with user-defined nonmenu maps:

- AGR-DIALOG-NAME
- AGR-APPLICATION-NAME
- AGR-CURRENT-FUNCTION
- AGR-FUNCTION-DESCRIPTION
- AGR-DATE
- AGR-USER-ID
- AGR-MODE
- AGR-PASSED-DATA
- AGR-MAP-RESPONSE

For an illustration of how these fields can be used on maps, refer to "Designing maps" in Chapter 4.

Chapter 4. Naming Conventions

4.1 Overview

The establishment of naming conventions reduces the accumulation of redundant data and improves the overall design of an application. Naming convention standards apply to the components of an application as well as to the database entities accessed by the application. Naming conventions for application entities and database information entities are each discussed separately below.

4.2 Naming application entities

Naming conventions make it easier to keep track of application components as they are created and maintained. While mnemonic names can work well for less complex applications, mnemonics are inadequate when handling the large volume of complex applications that typically exist at most sites. Adhering to a naming convention eases the construction of component names, eases the reconstruction of component names if one is forgotten, and eases the use and maintenance of an application.

Naming convention standards: The following table lists the naming convention standards used for the sample application in this manual.

Position	Value	Meaning
1	C	CA-IDMS product
2-3		Type of application:
	EM	Employee information
	IS	Information system
	FS	Financial system
	MS	Manufacturing system
	SY	System activities
4		Component type:
	D	Dialog
	F	Function
	M	Map
	P	User-defined program
	H	Help module
	R	Report
	S	Report
	T	Subschema
	U	Table
		Menu
5		Component functions:
	A	Add operation
	C	Encode/decode (column 4 indicates table)
	D	Delete operation
	E	Edit operation (column 4 indicates table)
	I	Inquiry operation
	M	Modify operation
	U	Update operation
6-8		Component designator:
	xxx	Three characters used as unique identifiers for the component

Assigning names: Names in an application can be assigned in the following manner:

- Dialogs, maps, tables, programs, help modules, and reports can use the conventions in preceding table as follows:

`Dialog:`	`CEMDILIS`
`Map:`	`CEMMILIS`
`Code table:`	`CEMTCLIS`
`Edit table:`	`CEMTELIS`
`Menu:`	`CEMUILIS`
`User program:`	`CEMPILIS`
`Help module:`	`CEMHILIS`
`Report:`	`CEMRILIS`

- Dialog premap and response process names can be the concatenation of the dialog name and the suffix -PREMAP or -RESPONSE, as in the following examples:

  ```
  CEMDILIS-PREMAP

  CEMDILIS-RESPONSE
  ```

  If there are multiple response processes, the suffixes can be structured to reflect the function of each response process, as follows:

  ```
  CEMDILIS-ADDRESP
  CEMDILIS-DELRESP
  ```

- Names for subroutines included in the premap and response processes can be made up of a meaningful name of up to 6 characters with a 2-digit suffix, as follows:

  ```
  PASSDT05
  MESSGE97
  DBERR99
  ```

  The numeric suffixes can be assigned and incremented as the subroutines appear in the dialog. This numbering convention makes it easier to locate a subroutine in the dialog listing. For example, MESSGE97 is located near the end of the listing while PASSDT05 is located near the beginning.

4.3 Naming database information entities

Glossary: The creation of a glossary can be an effective means of establishing naming conventions for database information. The glossary can be stored in the dictionary where it is readily available as a reference tool. Tools such as the glossary also aid in the development of consistent site-specific application coding standards.

Example: The following example shows sample entries from one type of glossary. This example shows one way in which a glossary can be defined; each design team must determine the naming conventions that best suit its needs. Note that the word WORD in this example is a user-defined entity defined to the data dictionary, as follows:

The sample entries from this glossary show one way in which naming conventions can be implemented within an installation. In this glossary, the application designers have determined that certain words are always to be abbreviated and others are never to be abbreviated; the majority of words are to be spelled out completely whenever possible. When stored on the data dictionary, the glossary is readily available as a reference guide for programmers and developers.

```
ADD CLASS NAME IS WORD
CLASS TYPE IS ENTITY.



ADD WORD ABEND          ABBREVIATED NEVER
ADD WORD ABSOLUTE       ABBREVIATED NEVER
ADD WORD ACCEPT         ABBREVIATED NEVER
ADD WORD ACCOUNT        ABBREVIATED SOMETIMES ABBR ACCT
ADD WORD ACCRUAL        ABBREVIATED NEVER
ADD WORD ACCUMULATE     ABBREVIATED SOMETIMES ABBR ACCUM
ADD WORD ACKNOWLEDGE    ABBREVIATED SOMETIMES ABBR ACK
ADD WORD ADMINISTRATION ABBREVIATED ALWAYS ABBR ADMIN
ADD WORD ADDRESS        ABBREVIATED ALWAYS ABBR ADDR
     .
     .
     .
     .
     .
     .
ADD WORD YIELD          ABBREVIATED SOMETIMES ABBR YLD
ADD WORD YTD            ACRONYM 'YEAR TO DATE'
ADD WORD YY             ABBREVIATED NEVER
ADD WORD ZERO           ABBREVIATED NEVER
ADD WORD ZONE           ABBREVIATED NEVER
```

Information entities can use the following naming conventions:

- Database elements can be established using approved names from the glossary and can be further defined with synonyms. Element names should have a maximum of 25 characters. The following example lists an element and three synonyms:

```
EMPLOYEE-CODE

DB-REC-EMPLOYEE-CODE
MAP-EMPLOYEE-CODE
WORK-EMPLOYEE-CODE
```

- Database Records can be composed of approved, usable names (for example, EMPLOYEE). Records can be given greater flexibility with the addition of suffixes. The following example lists employee records with identifying suffixes:

  ```
  EMPLOYEE-0600
  EMPLOYEE-2500
  EMPLOYEE-6359
  ```

  In CA-ADS process source, as well as in COBOL, CA-CULPRIT, and map source, the elements can be referenced by the element name plus the suffix,as follows:

  ```
  EMPLOYEE-CODE-6359
  ```

- Map work records are composed of the map name followed by the suffix -MAP-RECORD, as in the following example:

  ```
  CEMMILIS-MAP-RECORD
  ```

  Elements in the map record utilize the prefix MAP- and the element name, as follows:

  ```
  MAP-OFFICE-CODE
  ```

  If the map needs more than one work record, a number is added to the word MAP, as follows:

  ```
  CEMMILIS-MAP2-RECORD (the second map record)

  MAP2-OFFICE CODE (a record element from the second record)
  ```

- Dialog work records are composed of the dialog name followed by the suffix -WORK-RECORD as in the following example:

  ```
  CEMDULIS-WORK-RECORD
  ```

  Elements in the dialog work record utilize the prefix WORK- and the element name, as follows:

  ```
  WORK-OFFICE-CODE
  ```

  If the dialog needs more than one work record, a number is added to the word WORK, as follows:

  ```
  CEMDILIS-WORK2-RECORD (the second dialog work record)

  WORK2-OFFICE CODE (a record element from the second record)
  ```

- Set names are established by concatenating an abbreviation of the owner record (a 7-character maximum) with that of the member record (a 6-character maximum), as follows:

  ```
  EMPL-SKILL
  ```

Chapter 5. Performance Considerations

5.1 Overview

The performance of the CA-ADS runtime system is dependent upon a number of factors, such as the size of the CA-IDMS/DC or CA-IDMS/UCF (DC/UCF) system, the number of applications being run concurrently, and the number of users for a given application. Rather than attempting to give definitive instructions for the improvement of performance, this section discusses the following aspects of the CA-ADS runtime system:

- Parameters affecting performance
- Resource management

Each of these considerations is discussed separately below.

5.2 System generation parameters

The CA-ADS system is generated by submitting ADSO, PROGRAM, and TASK statements to the DC/UCF system generation compiler. Optionally, the KEYS statement is used to define site-specific control key functions. These statements are used as follows:

- The ADSO statement includes parameters that define the CA-ADS runtime environment, as follows:
  - The task code (ADS) that initiates the CA-ADS runtime system
  - The mainline dialog that can begin executing immediately
  - The maximum number of dialog levels that can be established by each application
  - The disposition of record buffers during a pseudo-converse (that is, whether they can be written to the scratch area of the data dictionary)
  - The size of the primary and secondary record buffers
  - The AUTOSTATUS facility that handles errors generated by database, logical record, or queue and scratch record processing
  - The Status Definition Record that associates status codes returned by database, logical record, and queue and scratch record access with condition names
  - The treatment of numeric values placed into alphanumeric fields by arithmetic and assignment commands
  - The display of the Dialog Abort Information screen when the runtime system detects an abend condition in an executing dialog
  - Whether dialog statistics will be collected
  - How CA-ADS is to perform a mapout when a dialog's map is already displayed as a result of a previous mapout in a pageable map
  - Whether record buffer blocks are to be compressed across a pseudo-converse when they are retained in the storage pool
  - How the amount of storage to be allocated for record buffer blocks is to be determined
- The PROGRAM statement defines the following CA-ADS components as DC/UCF programs:
  - The ADSORUN1, ADSORUN2, and ADSOMAIN runtime system programs
  - The system maps (the menu map, runtime message map, and maps for each of the application and dialog compiler screens)
  - The application and dialog compiler programs (ADSA and ADSC)
  - CA-ADS dialogs (an optional parameter if null Program Definition Elements (PDEs) are defined in the SYSTEM statement)

- The TASK statement defines the following task codes:
  - ADS and ADS2 to initiate the runtime system
  - ADSA to initiate the application compiler
  - ADSC to initiate the dialog compiler
  - ADSR to initiate the runtime system when returning from a linked user program

For detailed syntax and examples of the sysgen statements, refer to the *CA-IDMS System Generation* manual.

The following discussion highlights selected aspects of system generation that have particular import when considering system performance in an CA-ADS environment. These features are as follows:

- Allocating primary and secondary storage pools
- Relocating resources
- Specifying the number of online tasks and external request units

Each of these considerations is discussed separately below.

5.2.1 Allocating primary and secondary storage pools

Record Buffer Block: The runtime system allocates and initializes record buffers for use by executing dialogs. When an application is initiated, CA-ADS allocates a Record Buffer Block (RBB) from the DC/UCF storage pool to hold the subschema, map, and work records accessed by the dialogs in the application thread. The RBB must be large enough to accommodate the largest of these records.

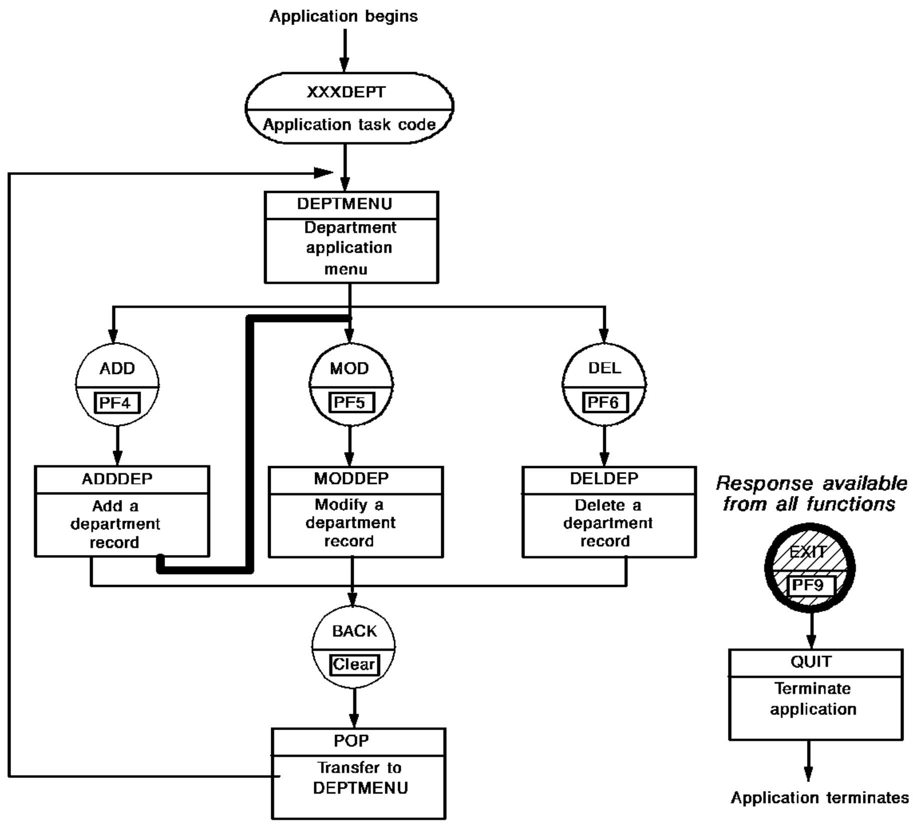
There is one primary RBB for each application. CA-ADS allocates a secondary RBB when the RBB becomes full during execution of the application or does not have enough remaining space to hold a record. Additional secondary RBBs can be allocated by the CA-ADS runtime system as necessary.

The data communications administrator (DCA) can specify the size of the primary and secondary RBBs with the PRIMARY POOL and SECONDARY POOL parameters of the ADSO statement. When allocating the primary and secondary storage pools, the DCA needs to consider the size and number of the records used by the application as well as the header records maintained by the buffers.

The primary RBB should be large enough to satisfy the records associated with the most-frequently used dialogs. The secondary RBB should be large enough to accommodate the largest record.

Alternatively, the runtime system can be directed to calculate the size of the RBBs for an application or dialog and to use the calculated size when acquiring storage space for the RBBs. Use the calculated size in systems where there is a high number of records and storage space is a concern.

The following diagram shows the structure of the Record Buffer Block. Each record buffer contains a 24-byte header to keep track of available space. For each record in the pool, CA-ADS maintains a record header (RBE) that requires 44 bytes of storage for database records and 56 bytes of storage for logical records. There is also a header for each element of a logical record. All records and headers are aligned on doubleword boundaries. Each buffer pool must be large enough to accommodate the largest subschema, map, work, database, or logical record used by a dialog in the application.



5.2.2 Relocating resources

The fast mode threshold is used by the CA-ADS runtime system in conjunction with the RESOURCES ARE FIXED specification to determine whether record buffers are written to disk or kept in main storage across a pseudo-converse. If the total size of all record buffers, in bytes, exceeds the fast mode threshold and RESOURCES ARE FIXED has been specified, the record buffers are written to disk; otherwise, the record buffers are kept in the storage pool. Storage used for currency blocks, CA-ADS terminal blocks (OTBs), OTB extensions, and variable dialog blocks (VDBs) is not eligible for writing to scratch.

Size of the threshold: The size of the threshold is a site-specific determination that is based on the availability of general resources versus the amount of available storage. I/Os for DC/UCF journaling and CPU cycles for record locking are used when record buffers are written to the scratch/queue areas. Therefore, when buffers exceed the fast mode threshold, the increased use of resources will slow down the transaction response time. On the other hand, if buffers are always under the threshold (that is, if the fast mode threshold is high), more memory is required.

Alternative method: Alternatively, the DCA can specify that storage used for RBBs and statistics control blocks is always written to scratch across a pseudo-converse, regardless of the relocatable threshold that has been defined for primary and secondary storage pools. In this situation, other storage (that used for currency blocks, OTBs, OTB extensions, and VDBs) is written to scratch across a pseudo-converse only when the relocatable threshold is exceeded.

5.2.3 Specifying the number of online tasks and external request units

The MAXIMUM TASKS and MAXIMUM ERUS parameters specify the maximum number of user tasks (online tasks) and external request units that can be active concurrently. The size of these parameters can affect the amount of time spent by DC/UCF in searching the queues for tasks that are waiting to be executed.

Considerations: The number of online tasks and external request units that should be specified is a site-specific determination and is dependent upon factors such as the number of tasks processed each hour in a particular environment. When setting the MAXIMUM TASKS and MAXIMUM ERUS parameters on the SYSTEM statement, the following statistics should be considered:

- Increasing the MAXIMUM TASKS or MAXIMUM ERUS parameters by one (1) causes virtual storage requirements to increase as shown below:

Resource	Size of resource	Total
TCE	248 bytes	248 bytes
STACKSIZE	320 words	1,280 bytes
DCE	40 bytes	40 bytes
ECB * 3	8 bytes	24 bytes
DPE * 20	16 bytes	320 bytes
RCE * 15	16 bytes	240 bytes
RLE * 25	8 bytes	200 bytes
Total increase per task		2,352 bytes

Note that a value larger than the default (420) should be specified for the STACKSIZE when using CA-ADS. If the STACKSIZE is at 420 and two tasks exceed stacksize and go into abend storage at the same time, the system will abort with an abend code of 3995.

- The following DC/UCF system parameters should be increased as specified for every increment of 1 in the size of MAXIMUM TASKS or MAXIMUM ERUS:

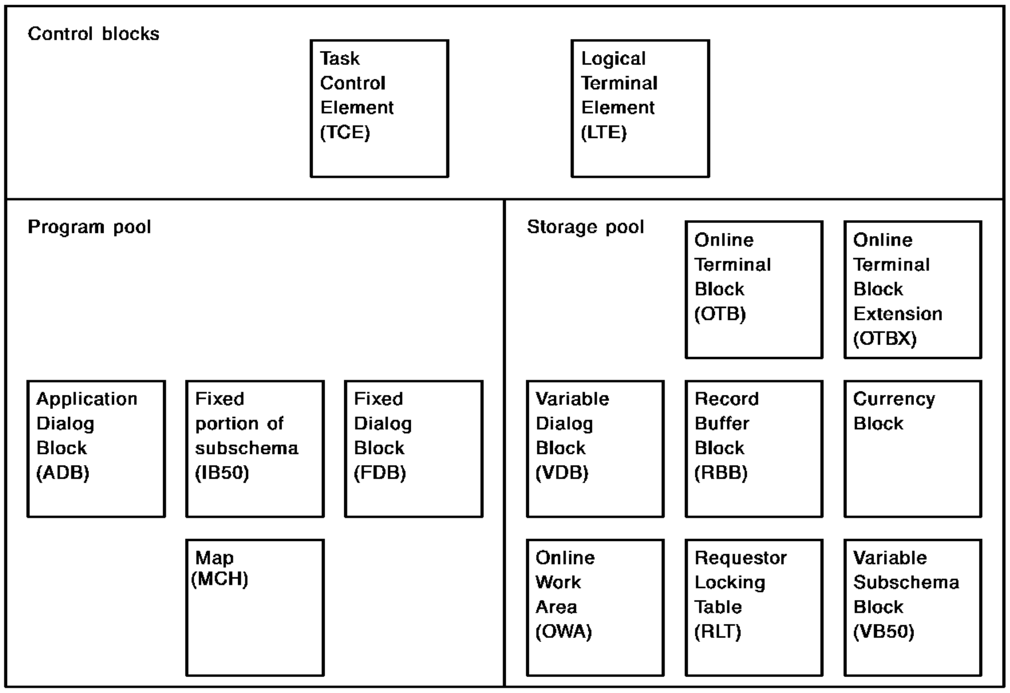
Parameter	**Amount increased**
ECB LIST	3
DPE COUNT	20
RCE COUNT	15
RLE COUNT	25

5.3 Resource management

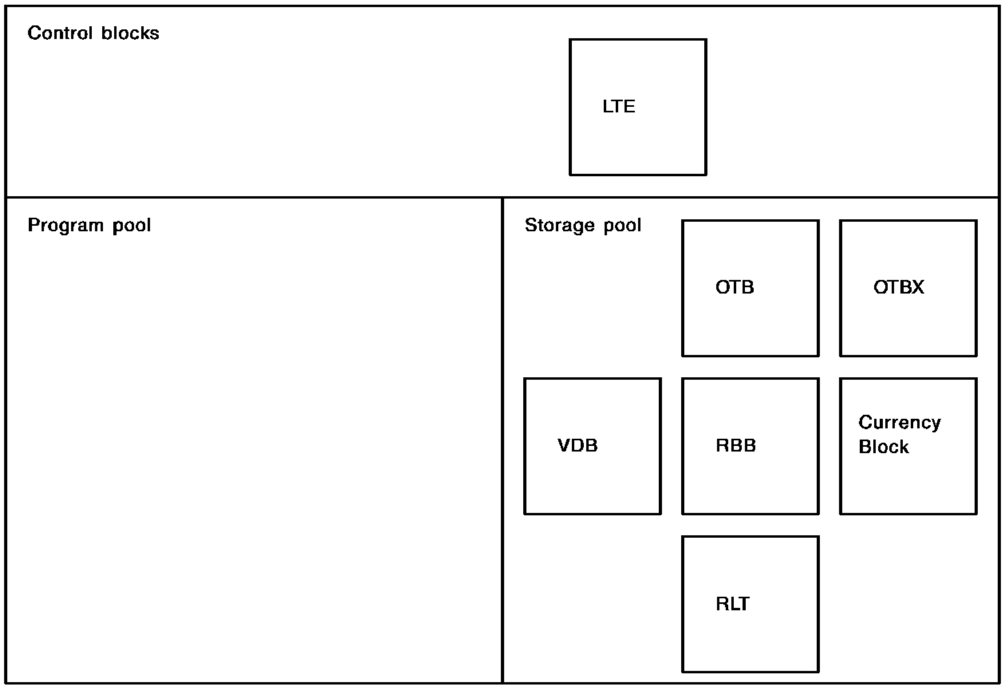
In designing applications, consideration must be given to the efficient management of system resources. The management of resources such as the database, the storage pool, and the program pool storage affects the performance of online applications since many users may require access to these resources simultaneously.

The following diagram shows the resources used by an application in a non-SQL environment while a task is active and after the task has terminated.

When a task is active:


Control blocks
Task Control Element (TCE)
Logical Terminal Element (LTE)
Program pool
Application Dialog Block (ADB)
Fixed portion of subschema (IB50)
Fixed Dialog Block (FDB)
Map (MCH)
Storage pool
Online Terminal Block (OTB)
Online Terminal Block Extension (OTBX)
Variable Dialog Block (VDB)
Record Buffer Block (RBB)
Currency Block
Online Work Area (OWA)
Requestor Locking Table (RLT)
Variable Subschema Block (VB50)

After the task has terminated:



The remainder of this section discusses methods that can be used to monitor the resource consumption of an application and ways in which to utilize available resources efficiently.

5.3.1 Monitoring tools

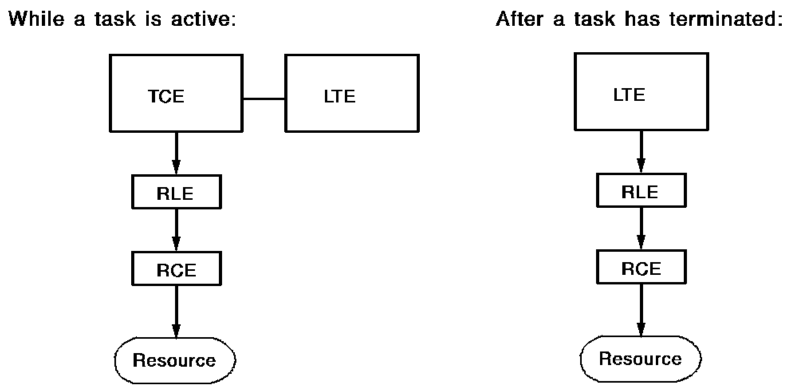
As with any task running under DC/UCF, the major resources to be monitored in an CA-ADS environment are as follows:

- Task processing support
- Variable storage pool
- Program pool storage
- Database locks
- I/Os (disk and terminal data transmission)
- CPU cycles

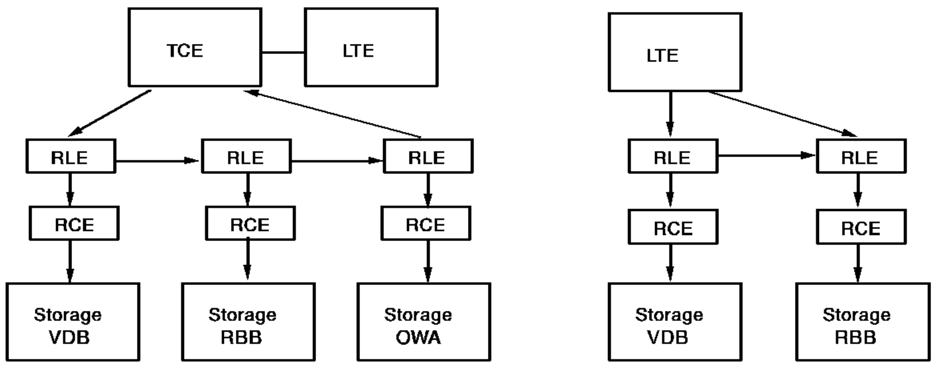
Each of these resources can be monitored with data dictionary reports and DC/UCF master terminal functions, as discussed below.

5.3.2 Task processing support

The following diagram shows the resources in use while a task is active (left diagram) and those in use after the task (right diagram) terminates.



Example:



Displaying internal resources: The following DC/UCF master terminal functions display the internal resources used to support task processing:

- DCMT DISPLAY ACTIVE TASK displays global statistics on active tasks and information on each active task thread.
- DCMT DISPLAY STATISTICS SYSTEM displays information about the system including the peak task control element (TCE) stack; and the maximum number of resource link elements (RLEs), resource control elements (RCEs), and deadlock prevention elements (DPEs) used by the tasks.

5.3.3 Variable storage pool

The following sysgen reports (CREPORTS) and DCMT functions can be used to monitor the use of the storage pool:

- CREPORT 25 verifies the size of the storage pool and indicates whether storage protection has been enabled for the system.

- DCMT DISPLAY ACTIVE STORAGE shows the current fragmentation of the storage pool.
- DCMT DISPLAY LTERM RESOURCES indicates which terminals are active and own resources.
- DCMT DISPLAY LTERM *lterm-id-a* **RESOURCES** displays the specific resources (and the addresses of those resources) owned by the named terminal.
- DCMT DISPLAY MEMORY *hex-address-a* displays an actual resource as it appears in memory.
- CREPORT 40 supplies the current parameters specified in the ADSO statement.

The *CA-IDMS Reports* manual describes CREPORTS; the *CA-IDMS System Tasks and Operator Commands* manual details the master terminal functions available to monitor system resources.

Information from the above displays and reports can be used to calculate the number of users the system can currently support, assuming various storage pool sizes.

5.3.4 Program pool storage

The following DCMT commands can be used to provide information on the program pool:

- DCMT DISPLAY ACTIVE PROGRAMS displays the following:
  - Statistics on program pool usage, including the total number of pages and total number of bytes in the pool; the number of loads to the program pool; the number of pages loaded; and the number of load conflicts
  - Information on currently active programs including the program name, type, and version number; count of users currently using the programs; size of the program in K bytes; the number of times the program was called; and the number of times the program was loaded into the program pool
  - The program pool page allocation map that shows which pages are not in use; which pages are in use by one program; and which pages are in used by more than one program
- DCMT DISPLAY ACTIVE REENTRANT PROGRAMS displays the above information for the reentrant program pool and the active reentrant programs. If no reentrant pool is defined, the standard program pool is shown.

5.3.5 Database locks

The DCMT DISPLAY RUN UNIT and OPER WATCH DB RUN UNITS commands can be used to show the number of database locks being requested for a particular run unit. The number of database locks maintained by a DC/DC system has considerable impact on CPU usage.

These locks are specified at sysgen time by the RULOCKS and SYSLOCKS parameters of the SYSTEM statement.

For a discussion of database locks, refer to the *CA-IDMS Database Design*. For information on factors to consider when preparing the SYSTEM statement, refer to the *CA-IDMS System Generation* manual.

5.3.6 Disk I/O

The following reports can be used for monitoring disk I/O:

- JREPORT 004 shows the average number of I/Os to disk for a given program.
- DCMT DISPLAY RUN UNITS or OPER W DB RU shows if any run units are waiting for a journal buffer (as indicated by a run unit status value of IUH). IUHs occur most frequently when the fast mode threshold is set too low.

For information on the JREPORTS (journal reports), refer to the *CA-IDMS Reports* manual.

5.3.7 Terminal I/O

Monitoring steps: The following steps can be taken to monitor terminal I/Os:

1. Run the mapping utility (RHDCMPUT) for a report on a specific map. This report will display a picture of the map and the attributes currently assigned to the map. The report will also indicate whether BACKSCAN is enabled for any mapping fields. If BACKSCAN is in effect and the NEWPAGE option on the ADSO statement has been selected, extraneous data from the previous mapout may be left on the screen when a map is redisplayed. It is advantageous to have NEWPAGE in effect, however, because this option increases runtime efficiency by reducing the number of data fields that need to be transmitted to the terminal. For more information about the NEWPAGE feature, refer to the *CA-ADS Reference*.
2. Use DCMT VARY PTERM *pterm-id* TRACE ALLIO FF to cause the datastream being transmitted to the terminal to be written to the log as well.
3. Use SHOWMAP *map-name* in conjunction with DCUF USERTRACE to cause the datastream of a particular map to be traced.
4. Use DCMT VARY PTERM *pterm-id* TRACE ALLIO OFF to turn off the trace, suppressing any further transmission of datastreams to the log.
5. Run the print log utility (RHDCPRLG) to show the actual trace. Specify the following parameters in the utility JCL:

   ```
   PRINT ALL FOR ALL
   FROM time ON date
   TO time ON date
   ```

Transmission times can be calculated by analyzing the length of the datastream.

5.3.8 CPU usage

To monitor CPU cycles and obtain CPU usage by task, the system can be instructed to collect task statistics. It is advisable not to request task statistics unless there is a demonstrated need as they require considerable overhead and generate a large volume of data. Task statistics are requested by specifying TASK STATISTICS WRITE or TASK STATISTICS COLLECT on the SYSTEM statement. The statistics are written to the DC/UCF log.

>> For detailed information on collecting task statistics, refer to *CA-IDMS System Operations*.

5.3.9 Conserving resources

Resources can be conserved as follows:

- Enable storage protection — Storage protection is enabled by specifying PROTECT in the SYSTEM statement at system generation. The benefits of using storage protection are as follows:

  – CPU overhead is reduced because there are shorter chains for the system to walk.

  – Resources are clustered.

  To avoid SVC overhead, it is advisable to enable storage protection (that is, specify PROTECT) on the SYSTEM statement and to disable storage protection (that is, specify NOPROTECT) on the PROGRAM statement.

- Specify buffer sizes in multiples of 4084 bytes — The 4084-byte limit represents a multiple of 4K (4096 bytes) less the 12 bytes for pointer information and task ID address, as shown below:

RCE access	Task ID	Actual storage	RCE address
4 bytes	4 bytes	4084 bytes	4 bytes

  If a 4K page were selected, storage would have to be taken from two contiguous pages. The benefits of placing a 4084-byte limit on the amount of storage acquired are as follows:

  – Fragmentation of the storage pool is reduced when only one page is requested. Space is allocated in contiguous frames for a particular request. It is easier for the system to find one page rather than two contiguous pages.

  – Less CPU overhead is required because partial pages do not have to be calculated or scanned. When the system finds a request for a multiple of 4K (less the pointer information), it will immediately scan the pool looking for entirely empty pages, thus saving overhead.

- Limit the size of subschemas — Subschemas should be specified to the requirements of the application. The size of the currency block is directly related to the storage requirements of the variable subschema storage block (VB50) used at runtime; the runtime system maintains currency tables for every record, set, and area accessed by the dialog. Therefore, it is worthwhile to make subschemas as streamlined as possible.

- Limit the number of dialog levels — The MAXIMUM LINKS parameter of the ADSO sysgen statement specifies the maximum number of dialog levels that can be established by each respective CA-ADS application; keep this parameter low. A well designed application has as few levels as possible. The number of levels should be limited because, for each level established in the application, kept storage is acquired for the Variable Dialog Block (VDB) and the currency block. Storage established at a particular level is not released until control is passed upward.

  To limit the number of levels established, use the TRANSFER command whenever possible; build the application horizontally (that is, pass control laterally) rather than vertically.

- Control the size of the application — The size of dialog premap and response processes, the number of data fields included in a map, the size of the subschemas, and the size of database, map, and work records affect the performance of the CA-ADS runtime system. The actual number of I/Os required to load a complete program is dependent upon the size of a page in the DDLDCLOD area, the amount of overflow that will be encountered to load that record, and the size of the actual program being loaded. Therefore, the following benefits are realized by minimizing the size of programs:

  – A reduction in the work required to load a small program as compared to a large program

  – A reduction in time spent loading a particular program in the program pool or reentrant pool

  – A reduction in time spent waiting for space in the program pool or reentrant pool

  Under DC/UCF, the term program refers to CA-ADS dialogs, tables, maps, subschemas, and online and batch programs.

- Make frequently called programs resident — A frequently called program (such as ADSOMAIN) is virtually a resident in the program pool or the reentrant pool. The program should be made resident because the operating system can page more rapidly than IDMS-DC/UCF can read in a page from the DDLDCLOD area. By making the program resident, the operating system, rather than DC/UCF, will be requested to bring the page in core. Additionally, the program and resident pool will be less fragmented when a frequently used program is made resident. A program can be specified as resident on the PROGRAM statement at system generation.

- Free the resources of an inactive terminal — The resource timeout facility can be activated on the SYSTEM statement at system generation, specifying the amount of time a terminal is permitted to be inactive (that is, have no task executing) before all resources owned by the terminal are deleted and control is returned to the system. Because longterm storage resources are associated with a terminal even though a program is not active, freeing those resources will free space for other users of the system. This is particularly important if longterm locks are being implemented.

Chapter 6. Overview of CA-ADS Application Development

6.1 Introduction

Computer Associates' Application Development System (CA-ADS) enables you, as an application developer, to develop and execute online applications that query and update an CA-IDMS/DB database or VSAM file.

Development tools: The CA-ADS application development environment features **menu-driven development tools** that simplify and speed the definition of applications. Components developed by using CA-ADS development tools are modular and can be reused in one or more applications.

Process language: To enable an application to access the database, perform calculations, and perform other customized processing, you use the **high-level process language** provided by CA-ADS. Easily recognizable commands (such as DISPLAY) greatly simplify many frequently performed operations. Modules written in this process language can be added to an application whenever processing is required.

CA-ADS is fully integrated with Computer Associates' Application Development System/Batch (CA-ADS/Batch). Developers who use CA-ADS/Batch to define batch applications can use the same development tools used by CA-ADS developers.

>> For more information on CA-ADS/Batch, see the *CA-ADS Batch User Guide*.

Additionally, CA-ADS applications can accommodate components developed by using the CA-IDMS Automatic System Facility (ASF). Execution of CA-ADS applications can be traced and debugged by using the CA-IDMS online debugger or CA-ADS/ALIVE. Runtime performance and resource usage of an application can be tracked by using the CA-IDMS Performance Monitor.

This chapter introduces CA-ADS by discussing:

- **Application development** in the CA-ADS environment
- **Application development tools** that will be discussed in this user's guide

6.2 Application development

Flexible application development environment: CA-ADS provides a flexible application development environment. You can define application components in whatever order makes the most sense given the application you need to define. For example, you can define the executable structure, screen displays, and runtime flow of control for the entire application before you define any modules of process code for the application. Alternatively, you can fully define all components, including modules of process code, for a subset of the application before even beginning to define components for the remaining application.

Methodology: This manual illustrates CA-ADS concepts by showing you how they apply to the definition of a sample application. The methodology presented here shows you one way to efficiently develop a new application by using CA-ADS development tools. At your site, other application development strategies may be employed.

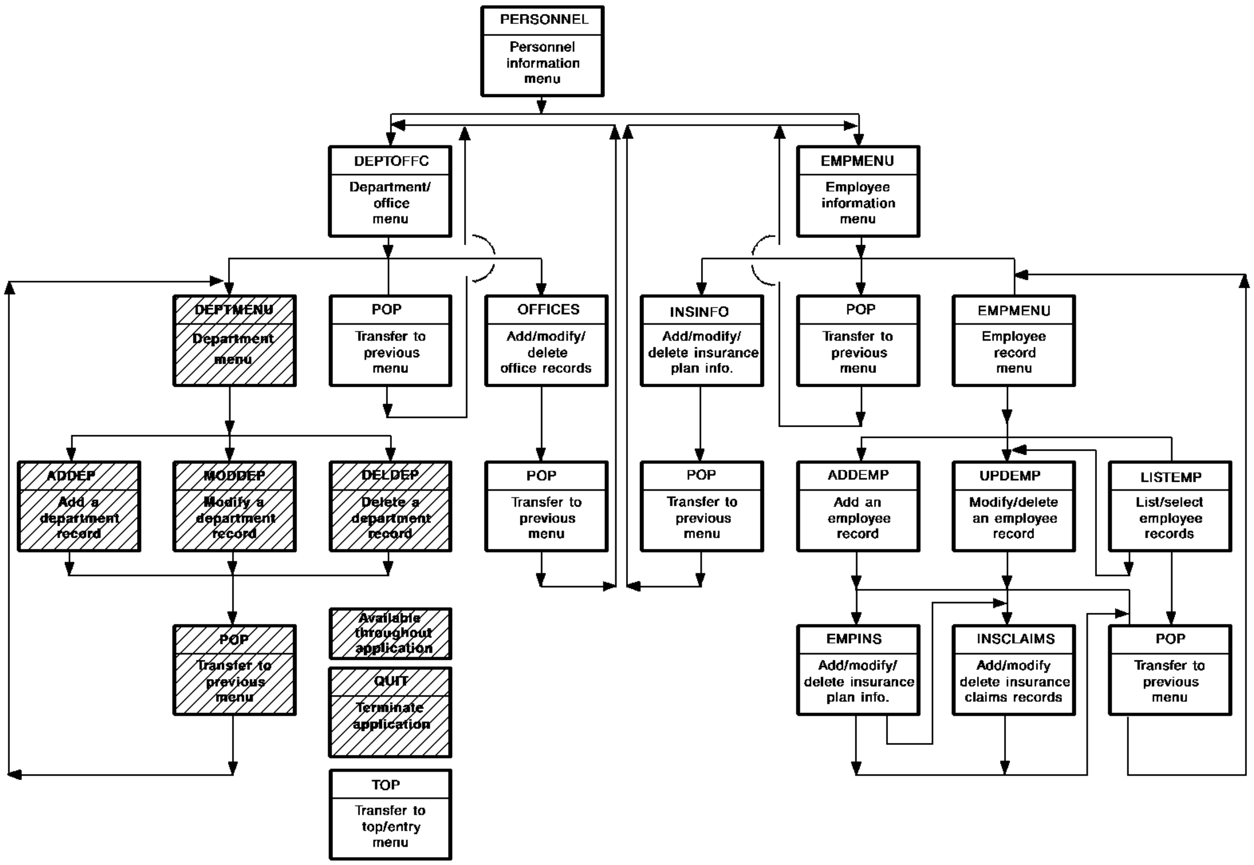
Structure diagram: A useful first stage in developing a new CA-ADS application is to develop a **structure diagram** based on the specifications for the application. Drawing a structure diagram for an application is useful because:

- **The application's functional requirements are clarified before any specific development occurs**. Potential design and development misunderstandings are often caught in the process of developing a structure diagram.
- **The structure diagram is a development resource for application developers throughout the development cycle**. While developing application components, the developer can use the structure diagram as a reminder of how each component relates to the entire application. CA-ADS application components fit directly into an application structure diagram.

Sample application: The structure diagram for a small Personnel application is shown below. This application allows users to maintain information on departments and users. Related data, such as office addresses and insurance information, is also collected and stored by using this application. You will develop the Department portion of this application in this manual.

Functions: As shown in this sample structure diagram, the application structure is made up of **functions**. Each function is related to work performed by the end user on a single screen. For example, the Personnel application includes functions such as ADDDEP, MODDEP, and DELDEP, which allow you to add, modify, and delete department records in the database. Menu functions, such as MAINMENU and DEPTMENU, allow you to select other functions in the application. Other functions allow you to return to previous menus (functions BACK and TOP) and to exit from the application (function EXIT).

To reduce repetition of information, this manual presents steps for developing only the Department portion of this sample Personnel application. Remaining portions of the Personnel application can be added to the Department application at any time. The


PERSONNEL
Personnel information menu
DEPTOFFC
Department/ office menu
EMPMENU
Employee information menu
DEPTMENU
Department menu
POP
Transfer to previous menu
OFFICES
Add/modify/ delete office records
INSINFO
Add/modify/ delete insurance plan info.
EMPMENU
Employee record menu
ADDEP
Add a department record
MODDEP
Modify a department record
DELDEP
Delete a department record
ADDEMP
Add an employee record
UPDEMP
Modify/delete an employee record
LISTEMP
List/select employee records
EMPINS
Add/modify/ delete insurance plan info.
INSCLAIMS
Add/modify delete insurance claims records
Available throughout application
QUIT
Terminate application
TOP
Transfer to top/entry menu

strategy of defining different portions of applications at different times is particularly useful when developing large applications.

Defining the application: As soon as the application structure diagram is completed and approved, you can begin to define the application in the data dictionary. In this manual, you will define the sample Department application in two steps:

1. **You will develop a prototype application** based on the approved structure diagram for the application.

   The CA-ADS runtime system simplifies the definition of prototypes by automatically performing many basic processing activities, such as displaying a screen. You can define a prototype without writing any process logic. Additionally, you can execute a prototype application before the database is developed.

2. **You will enhance the tested and approved prototype** by adding process logic.

Advantages: Advantages to developing a prototype application before defining any process logic for the application include:

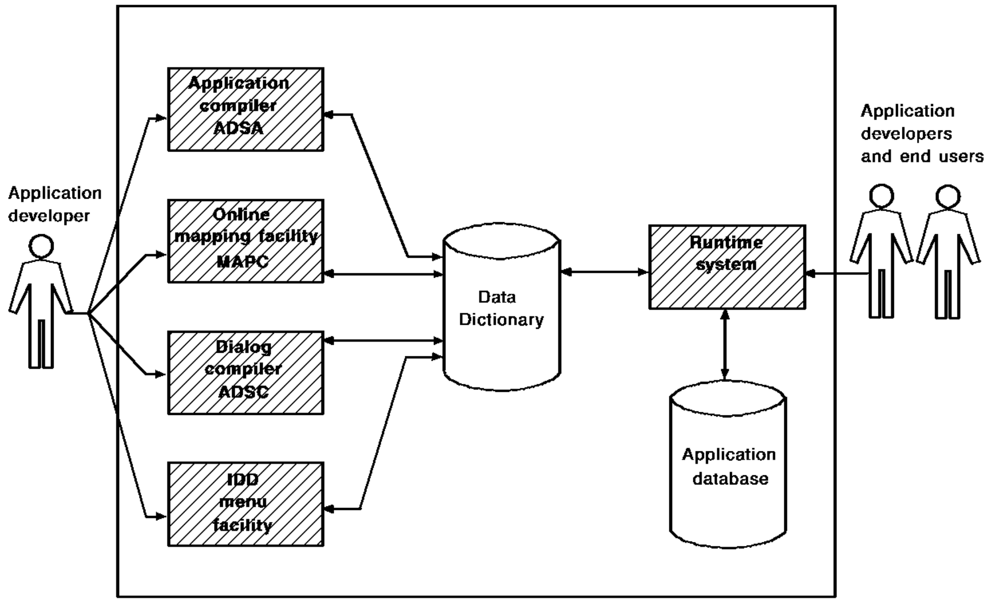
- **Timely feedback** — End users and other interested people can execute the completed prototype application to see how well screens meet their needs and to verify that flow of control from function to function makes sense based on their job responsibilities.

- **Ease of modification** — Improvements suggested by end users can be included in the prototype application quickly because you do not have to modify any process logic or database definitions.
- **Efficient use of development time** — The prototype application developed for early testing is itself developed into the final production application. Components created for the prototype are all used in the final application.

6.3 Application development tools

CA-ADS applications are developed and executed by using a variety of online tools. The following tools are used to develop the sample Department application.

- **CA-ADS application compiler (ADSA) —** Used to define the executable structure of an application, based on the structure diagram developed for the application.
- **CA-ADS dialog compiler (ADSC) —** Used to define CA-ADS **dialogs**, which handle most runtime interactions with the end user.
- **Online mapping facility (MAPC) —** Used to define **maps**, which establish preformatted screens. Dialogs use maps to display and allow end users to input information.
- **Integrated Data Dictionary (IDD) menu facility —** Used to create data definitions and modules of process code.
- **Runtime system —** Used to execute CA-ADS applications at any stage in the application's life cycle.



As an application developer, you can execute the application at any time in the development cycle by using the runtime system. Additionally, end users execute the application by using the runtime system.

Components defined by using ADSA, ADSC, MAPC, and the IDD menu facility are all stored in the data dictionary.

You can access any of the above development tools from CA-IDMS/DC, the CA-IDMS teleprocessing (TP) monitor, or CA-IDMS/UCF, the teleprocessing monitor interface. Additionally, you can transfer directly among ADSA, ADSC, MAPC, and the IDD menu facility by using the **transfer control facility (TCF)**.

Transfer control facility You can use TCF at any time during an application development session to suspend one development tool and transfer to another.**:** For example, while using ADSC to define a dialog, you might remember that the related map definition is still incomplete. You can suspend your ADSC session, transfer directly to MAPC to complete the map, and then transfer back to ADSC to resume your suspended dialog-definition session.

Task codes: To invoke a development tool from CA-IDMS/DC or CA-IDMS/UCF (DC/UCF), or TCF, specify the **task code** associated with the tool. A task code is a unique invocation name defined for a development tool at system generation time. Sample task codes for CA-ADS development tools are presented in the table below. The task codes shown allow you to operate under TCF and, therefore, switch from tool to tool.

Note: If you are not operating under TCF, you cannot switch to another tool without first returning to DC/UCF.

Task codes can vary from site to site.

Development tool	**Sample task code**	**Site task code**
CA-ADS application compiler (ADSA)	ADSAT	
CA-ADS dialog compiler (ADSC)	ADSCT	
IDD menu facility	IDDMT	
Online mapping (MAPC)	MAPCT	

Chapter 7. Defining an Application Structure Using ADSA

7.1 Introduction

As an application developer, you can begin defining application components as soon as the application structure diagram is complete. You can use CA-ADS development tools to define a preliminary, or **prototype** version of the application early in the application development cycle. Application developers and users can execute the prototype to test and suggest improvements to the application.

As soon as the prototype application is approved, definitions that make up the application can be enhanced to perform all processing required by the final production application. In this part of the manual, "Developing the Prototype," you will develop a prototype Department application. Then you will take this prototype application and develop it into a fully functional application.

The first step in defining a prototype application in the data dictionary is to define the application's structure, based on the structure diagram, by using the CA-ADS application compiler (ADSA). The structure of an application specifies how a user moves from function to function when executing the application.

In this chapter, you will begin defining the sample Department application introduced in Chapter 6, “Overview of CA-ADS Application Development” on page 6-1. This chapter includes:

- An overview of developing an application structure in the CA-ADS environment
- Instructions for defining the sample Department application
- A summary of what you've accomplished in this chapter

7.2 Overview

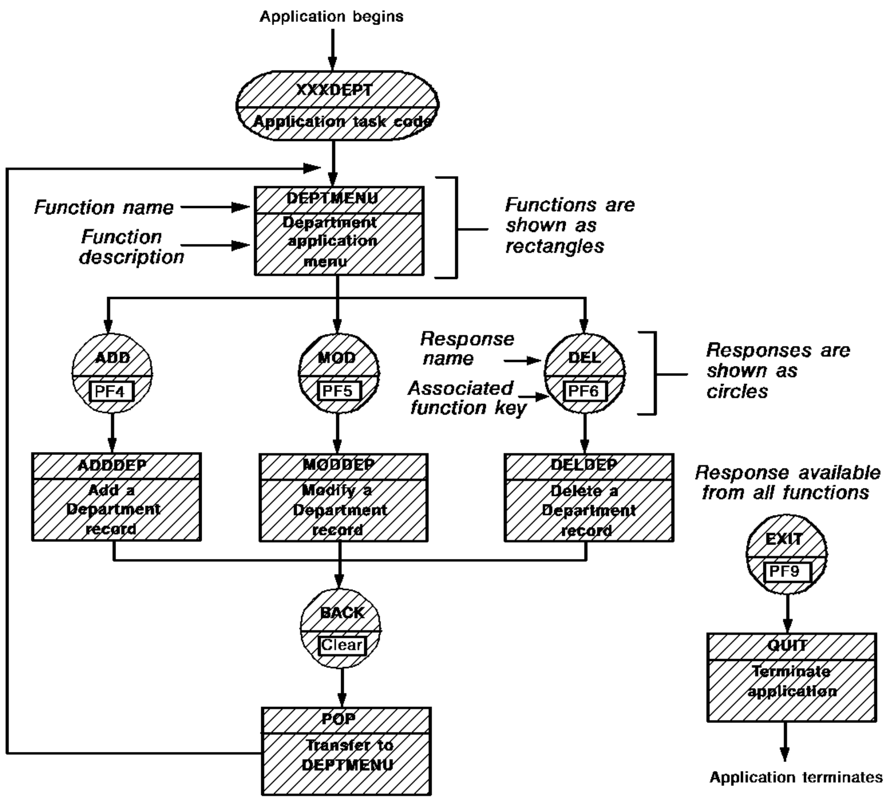
Functions: An application is made up of the various **functions** necessary to do the work of the application. Each function represents a unit of work in the application. For example, different functions in the Department application:

- Display a menu to help the user navigate through the application
- Perform functional processing, such as allowing the user to add or modify a department record
- Perform standard system activities, such as returning control to the application's main menu or exiting the user from the application

Responses: The development team must arrange application functions so that users can easily get from one function to another. The team defines paths that a user can take between functions. Each path is called a **response**. At runtime, the user can select from among available responses to get from one function to another.

Application structure: The arrangement of functions and responses constitutes the **structure** of an CA-ADS application. The diagram below shows the structure of the sample Department application. This diagram includes both functions and responses because you need to define both types of application components. The structure diagram for an application serves as a blueprint for application development. Functions and responses in this sample application will allow users to add, modify, and delete information about departments.

Note: In this sample application, you can substitute your initials for the *XXX* in the task code, application name, dialog names, etc.



Types of functions: Most functions in an CA-ADS application, including the Department application, are menu, dialog, or system functions.

Menu function: A **menu function** displays a menu screen to an end user.

For example, the DEPTMENU function in the Department application is a menu function. The menu screen displayed by DEPTMENU lists the ADD, MOD, DEL, and EXIT responses. At runtime, the user can select any of these responses from the DEPTMENU menu screen. To define a menu function, all you have to do is specify where the menu belongs in the application structure. CA-ADS provides predefined screen formats and processing logic for menu functions.

Dialog function: A **dialog function** typically displays a screen that supplies information to or requests information from a user. Based on entries made by the user, the function performs processing activities such as data retrieval and update. As a developer, you define CA-ADS programs, called **dialogs**, to handle runtime operations for dialog functions.

In the sample Department application, functions ADDDEP, MODDEP, and DELDEP are dialog functions. The user can use ADDDEP to add a new department record, MODDEP to modify an existing department record, and DELDEP to delete a department record from the database.

System function: A **system function** displays a screen and/or performs operations that are common to most applications.

For example, the POP system function in the sample Department application returns the user to the most recently used menu screen (in this case, the DEPTMENU menu). The QUIT system function allows the user to exit from the application at any point. To define a system function, all you have to do is specify where the function belongs in the application structure. CA-ADS provides all necessary logic and screens for system functions.

Paths between functions: **Responses** define paths between functions. At runtime, a user moves from one function to another by selecting a response that leads from the first function to the next. For example, a user of the Department application can get from DEPTMENU to ADDDEP by entering a nonblank in the selection field to the left of the ADD response on the menu.

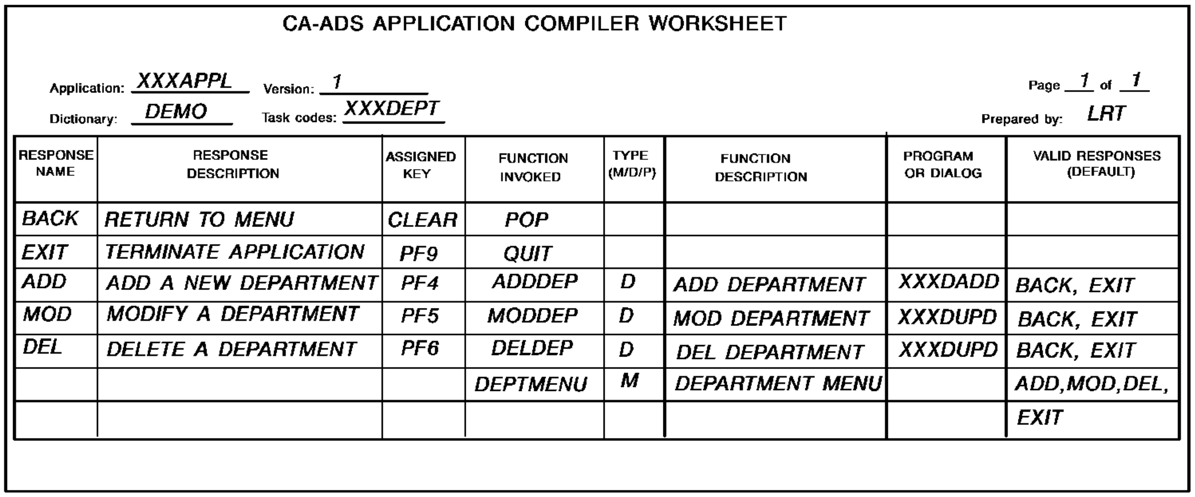
The user selects a response either by typing its name (for example, ADD) in a special response field on the screen or by pressing the function key (for example, [PF4]) associated with the response. Default responses can be defined for functions, making it easier for users to get to frequently used functions. The structure diagram shown above gives the name and function key for each response in the sample Department application.

Worksheets: A typical application has numerous responses and functions. Some responses and functions may be available in several places in the application. To help keep track of the responses and functions in an application structure, some sites develop a worksheet of responses and functions for the application. A standard worksheet that documents each response and function is especially useful when you are defining the application structure in the data dictionary. Filling out a worksheet for application responses and functions can simplify the application development process and can also provide documentation of application development decisions.

A sample worksheet for the Department application is shown below.

Task codes: To enable users to access an application, you define entry points into the application structure. You establish entry points for CA-ADS applications by defining **task codes**. When you define a task code, you associate it with a specific application function. At runtime, you use the task code to begin executing the application at the associated function. Both application developers and users invoke applications by using task codes.

For example, the *XXX*DEPT task code is associated with the DEPTMENU function in the sample Department application shown previously. A user can invoke the application from a CA-IDMS/DC or CA-IDMS/UCF (DC/UCF) system by specifying *XXX*DEPT and pressing [Enter]. The DEPTMENU menu screen is the first application screen shown to the user.



CA-ADS APPLICATION COMPILER WORKSHEET

Application: XXXAPPL Version: 1 Page 1 of 1

Dictionary: DEMO Task codes: XXXDEPT Prepared by: LRT

RESPONSE NAME	RESPONSE DESCRIPTION	ASSIGNED KEY	FUNCTION INVOKED	TYPE (M/D/P)	FUNCTION DESCRIPTION	PROGRAM OR DIALOG	VALID RESPONSES (DEFAULT)
BACK	RETURN TO MENU	CLEAR	POP				
EXIT	TERMINATE APPLICATION	PF9	QUIT				
ADD	ADD A NEW DEPARTMENT	PF4	ADDDEP	D	ADD DEPARTMENT	XXXDADD	BACK, EXIT
MOD	MODIFY A DEPARTMENT	PF5	MODDEP	D	MOD DEPARTMENT	XXXDUPD	BACK, EXIT
DEL	DELETE A DEPARTMENT	PF6	DELDEP	D	DEL DEPARTMENT	XXXDUPD	BACK, EXIT
			DEPTMENU	M	DEPARTMENT MENU		ADD,MOD,DEL,
							EXIT

Multiple task codes: You can define more than one task code for an application. Defining multiple task codes, or entry points, is particularly useful for very large applications. For example, if the sample Department application is incorporated into a large, company-wide application, users who only need to use department information can use the *XXX*DEPT task code to enter the application at the DEPTMENU function. Clerks in the Accounts Receivable office can use a different task code to enter the company-wide application at a menu that lists billing and customer functions.

Note: In the sample application, you can substitute your initials for the *XXX* in the task code.

7.3 Instructions

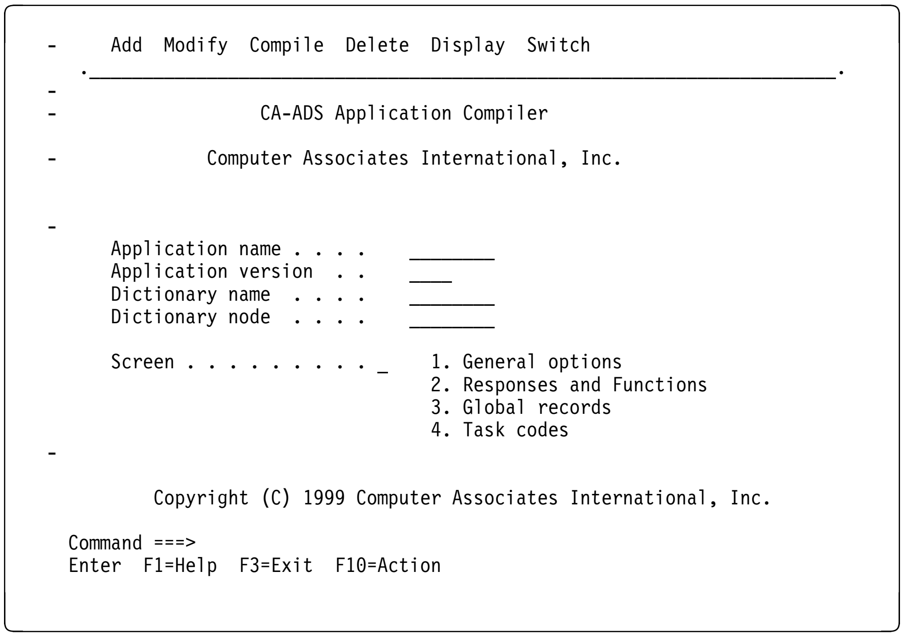
As a member of the development team, you use the **CA-ADS application compiler (ADSA)** to define an executable structure for an application. ADSA screens lead you through the steps necessary to define the functions, responses, and task codes that make up the application's structure.

When you have defined the application structure, you use ADSA to create a **load module** for the application. When you use ADSA screens to define the application structure, you are implicitly coding the flow of control for the application.

ADSA screens: ADSA screens are designed in accordance with the CUA definition of SAA. All screens have:

- A title at the top
- A data entry area
- PF keys at the bottom

Menu screens also have an action bar, a message area, and a command line.



```
-     Add  Modify  Compile  Delete  Display  Switch
   .______________________________________________________________________.
-
-                 CA-ADS Application Compiler

-            Computer Associates International, Inc.


-
      Application name . . . .    ________
      Application version  . .    ____
      Dictionary name  . . . .    ________
      Dictionary node  . . . .    ________

      Screen . . . . . . . . . _    1. General options
                                    2. Responses and Functions
                                    3. Global records
                                    4. Task codes
-

          Copyright (C) 1999 Computer Associates International, Inc.

  Command ===>
  Enter  F1=Help  F3=Exit  F10=Action
```

Steps: To define the sample Department application structure, you will perform the following steps:

1. Invoke ADSA
2. Name the application
3. Specify basic information about the application

4. Define application response and function relationships
5. Further define the application responses
6. Further define the application functions
7. Define the application task code
8. Compile the application (creating an application load module)

After you finish defining the Department application structure, you can exit from ADSA and optionally execute the application to test out the definitions that you have made. If you need additional information at any time about the use of ADSA, see "Using ADSA" in Appendix B.

Note: Only the menu access can be tested until prototype maps and dialogs are created.

The following diagram shows an overview of the flow and structure of ADSA.

Note: This overall structure applies to the MAPC and ADSC compilers as well as to ADSA.

7.3.1 Step 1: Invoke ADSA

To invoke ADSA, you must be signed on to a DC/UCF system. Procedures for signing on to DC/UCF differ from site to site. If you are not sure how to sign on, ask other users at your site for help.

Before you invoke ADSA, you may want to establish a default dictionary and/or Distributed Database (DDS) node for the current definition session under DC/UCF. Naming a **default dictionary** is useful if you are working in a multiple-dictionary environment. Naming a **DDS node** is useful only if you are working in a distributed database network.

At some sites, signon profiles for each user establish the appropriate default dictionary and/or node for each user that signs on to CA-IDMS/DC or CA-IDMS/UCF.

If you want to establish defaults for your current DC/UCF session, enter the appropriate commands, one at a time, after signing on to DC/UCF:

- You name a **default dictionary** by entering:

  ```
  DCUF SET DICTNAME dictionary-name
  ```

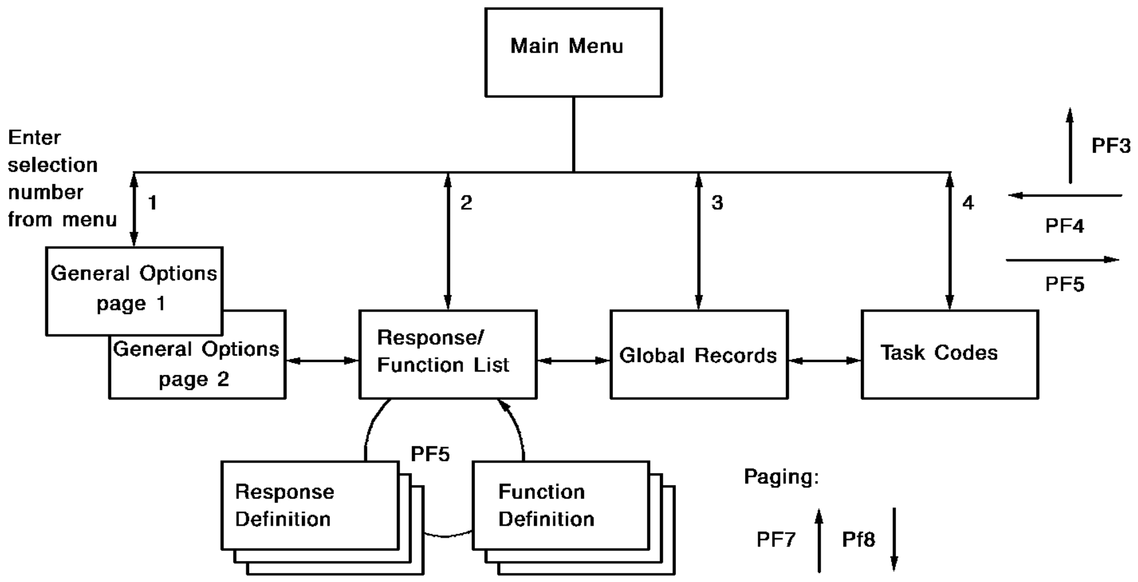
- You name a **default DDS node** by entering:

  ```
  DCUF SET DICTNODE node-name
  ```

You can display the names of your default dictionary and DDS node at any time by entering, one at a time:

```
DCUF SHOW DICTNAME
```

```
DCUF SHOW DICTNODE
```



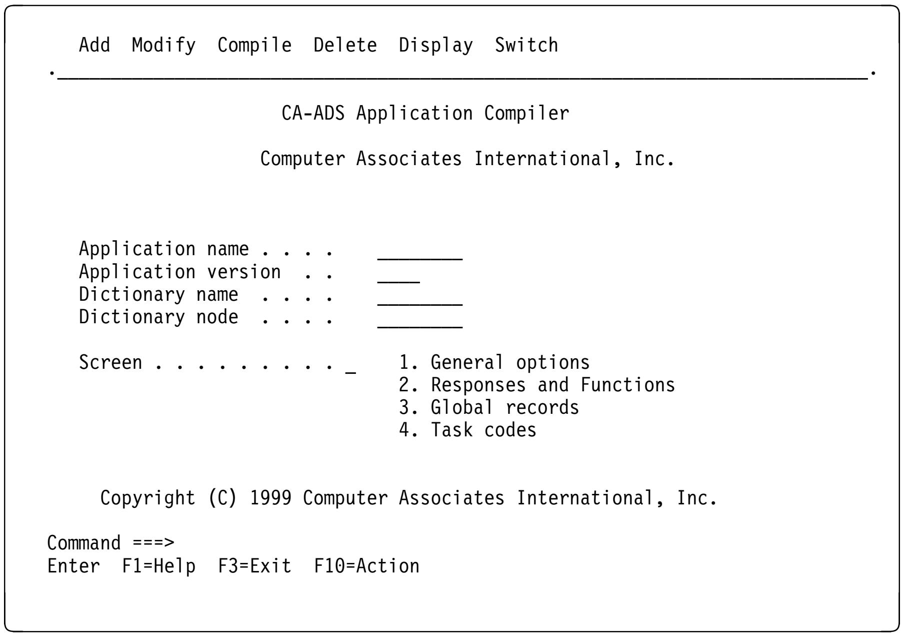
Specifying the task code for ADSA: When you've signed on to DC/UCF and optionally established a default dictionary name/node, you can invoke ADSA. To do this, you enter the task code for ADSA1.

For example, if the task code for ADSA is ADSAT, you can invoke ADSA from CA-IDMS/DC as shown:

```
ENTER NEXT TASK CODE:
adsat

                                        Press the ENTER
                                        key to input the --► [Enter]
                                        task code for
                                        ADSA.
```

Using the task code ADSAT means that you are invoking ADSA under TCF (the transfer control facility). Once you are in the ADSA tool under TCF, you can switch to another application development tool without returning to DC/UCF.

Main menu screen: ADSA begins with the Main Menu screen:



```
   Add  Modify  Compile  Delete  Display  Switch
._____________________________________________________________________________.
                     CA-ADS Application Compiler

                   Computer Associates International, Inc.



   Application name . . . .    ________
   Application version  . .    ____
   Dictionary name  . . . .    ________
   Dictionary node  . . . .    ________

   Screen . . . . . . . . . _    1. General options
                                 2. Responses and Functions
                                 3. Global records
                                 4. Task codes


     Copyright (C) 1999 Computer Associates International, Inc.

Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

If you are using a teleprocessing monitor other than CA-IDMS/DC, specify the task code for ADSA in response to the prompt presented by that monitor. For more information on task codes for CA-ADS development tools, see 6.3, “Application development tools” on page 6-7.

After you invoke ADSA, you can begin defining the application, as described below.

7.3.2 Step 2: Name the application

The ADSA **Main Menu** screen allows you to specify the name of the application, its version number, and the dictionary in which it resides. This dictionary name overrides the current default (if one has been set) and becomes the new default dictionary.

Screen prompts: When you begin an application definition, you typically enter information after one or more of the following Main Menu screen prompts:

- **Application name** — You must supply an application name. The name you specify must be unique among all program names (including applications, dialogs, maps, tables, help modules, etc.)
- **Application version** — You can optionally type a version number for the application after this prompt. When following the example in this manual, use the default version number of **1**.

  In a development environment, you might select a different test version number, such as 5. You would give all test components the same test version number. You would then set up the system so that, when you execute the application, test components are executed rather than production components.

  >> For more information on maintaining separate test and production definitions in the same data dictionary, see *CA-IDMS System Operations*.

- **Dictionary name** — A dictionary name may already be displayed after the **Dictionary name** prompt on your terminal. A sample dictionary name (DEMO) is shown in examples throughout this manual.

  If a dictionary name is not displayed, check with others at your site to find out if you need to specify one. You typically need to enter the name of your group's dictionary only when you are not using the default dictionary in a multiple-dictionary environment.

- **Dictionary node** — A dictionary node name may already be displayed after the **Dictionary node** prompt on your terminal. Sample definition screens in this manual do not show a node name.

  If a dictionary node is not displayed, check with others at your site to find out if you need to specify one.

1

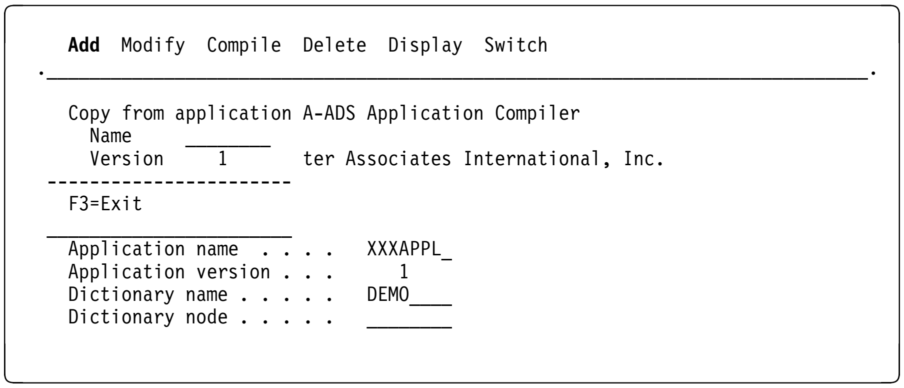
You can use the **tab key** to move the cursor quickly and easily between prompts. Begin defining your application on the Main Menu screen as shown below2:

```
   Add  Modify  Compile  Delete  Display  Switch
 .____________________________________________________________.

                    CA-ADS Application Compiler

                Computer Associates International, Inc.



    Application name . . . .    xxxappl_
    Application version  . .    1___
    Dictionary name  . . . .    demo____
    Dictionary node  . . . .    ________
```

Adding the application: To add the application, position the cursor on the **Add** item on the action bar and press [Enter]. You can position the cursor on **Add** by:

- Pressing [PF10] move to the action bar and then tabbing to **Add** and pressing [Enter]
- Tabbing to **Add** and pressing [Enter]
- Typing **add** on the command line and pressing [Enter]

1 2Sample input that you can enter on screens is shown in lowercase throughout this manual. On the sample Main Menu screen being shown here, the following sample input is shown in lowercase: a sample application name (XXXAPPL) and dictionary name (DEMO).



```
   Add  Modify  Compile  Delete  Display  Switch
 .______________________________________________________________________________.

   Copy from application A-ADS Application Compiler
     Name      ________
     Version      1      ter Associates International, Inc.
 -----------------------
   F3=Exit
 _______________________
   Application name  . . . .   XXXAPPL_
   Application version . . .      1
   Dictionary name . . . . .   DEMO____
   Dictionary node . . . . .   ________
```

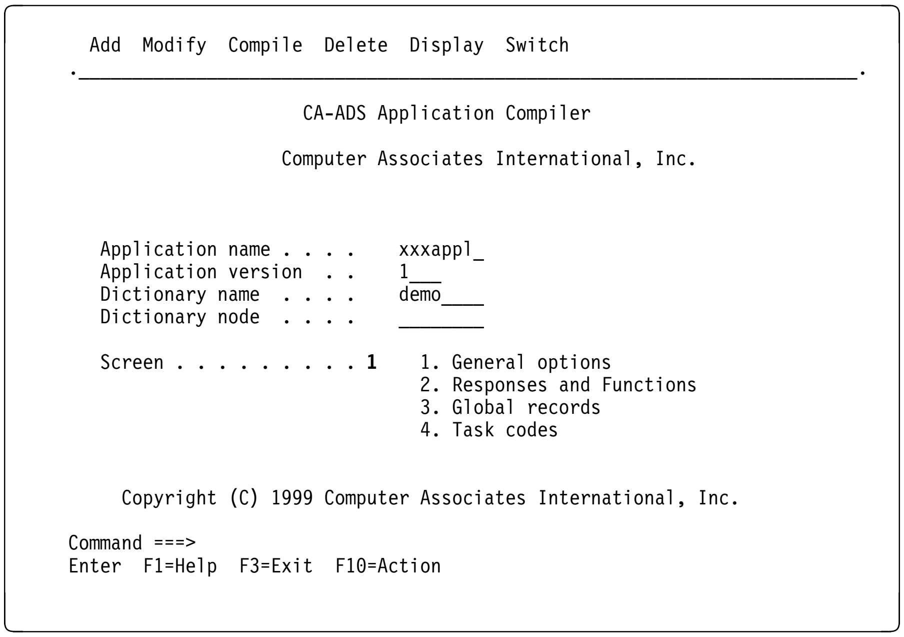
Once you have displayed the **Add** action item, press [Enter] to add the application to the dictionary. The action is confirmed.

ADSA redisplays the Main Menu screen with an appropriate message:

- A confirming message is returned when your definition contains no errors.
- An error message is returned if ADSA detects any errors. Read the message to see what problem has occurred. You can type over any fields in error and then press [Enter] again.

7.3.3 Step 3: Specify basic information

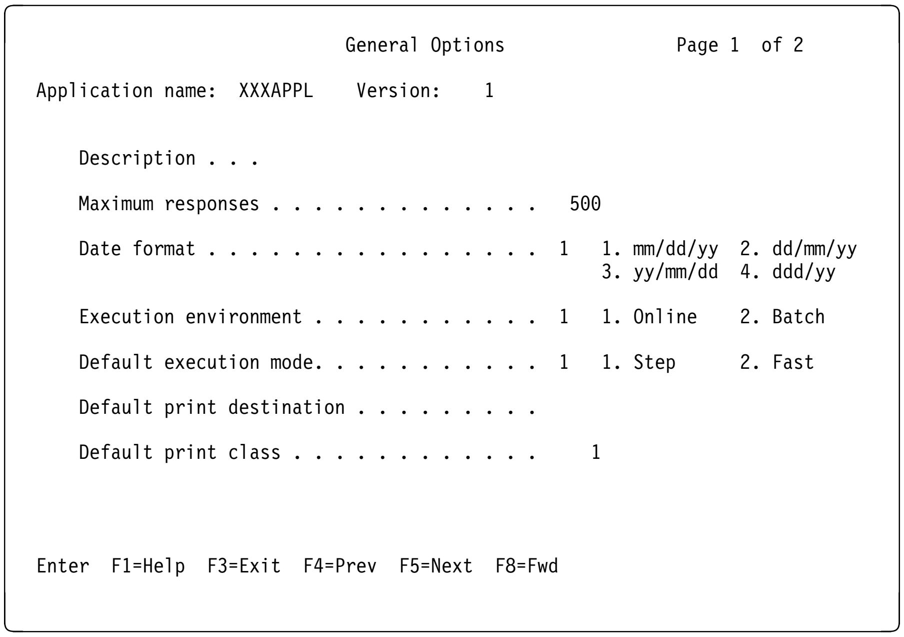
After you specify the name of the application, you can give some basic information about the Department application on the **General Options** screen. You reach the General Options screen by entering **1** next to **Screen** on the Main Menu.



```
     Add  Modify  Compile  Delete  Display  Switch
   .__________________________________________________________________________.

                        CA-ADS Application Compiler

                      Computer Associates International, Inc.


      Application name . . . .    xxxappl_
      Application version  . .    1___
      Dictionary name  . . . .    demo____
      Dictionary node  . . . .    ________

      Screen . . . . . . . . . 1    1. General options
                                    2. Responses and Functions
                                    3. Global records
                                    4. Task codes


        Copyright (C) 1999 Computer Associates International, Inc.

   Command ===>
   Enter  F1=Help  F3=Exit  F10=Action
```

The General Options screen is displayed.

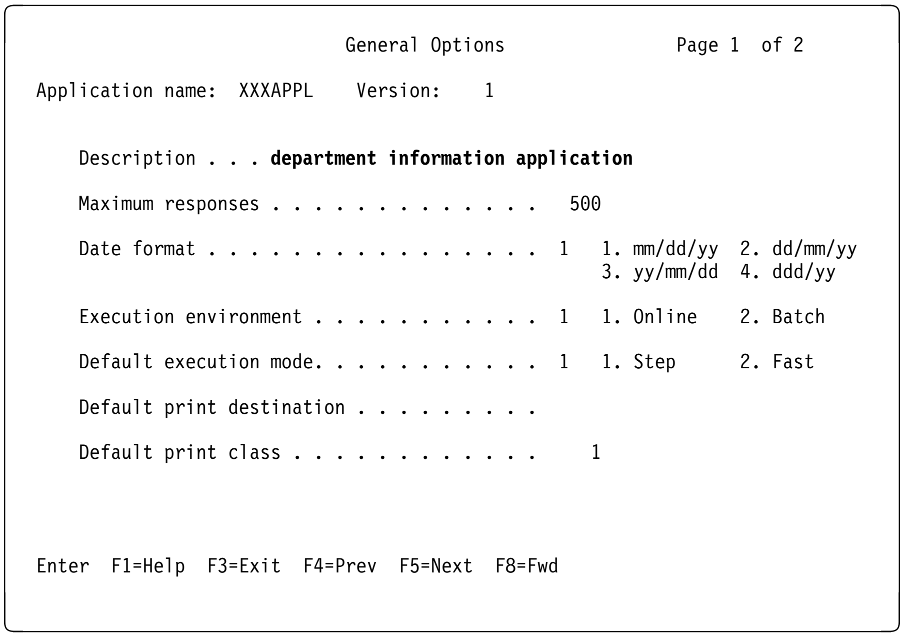
The General Options screen



```
                             General Options              Page 1  of 2

Application name:  XXXAPPL    Version:    1


    Description . . .

    Maximum responses . . . . . . . . . . . . .   500

    Date format . . . . . . . . . . . . . . . .  1   1. mm/dd/yy  2. dd/mm/yy
                                                     3. yy/mm/dd  4. ddd/yy

    Execution environment . . . . . . . . . . .  1   1. Online    2. Batch

    Default execution mode. . . . . . . . . . .  1   1. Step      2. Fast

    Default print destination . . . . . . . . .

    Default print class . . . . . . . . . . . .     1



Enter  F1=Help  F3=Exit  F4=Prev  F5=Next  F8=Fwd
```

Screen prompts: You typically enter a description on the General Options screen:

- **Description** — You can optionally type a one-line description of your application after this prompt.



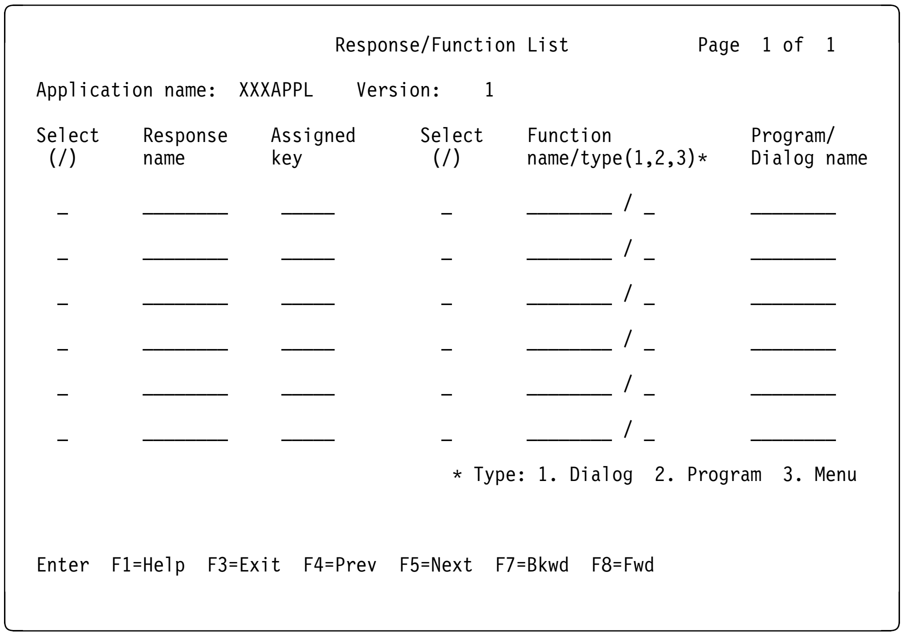
```
                             General Options              Page 1  of 2

Application name:  XXXAPPL    Version:    1


    Description . . . department information application

    Maximum responses . . . . . . . . . . . . .   500

    Date format . . . . . . . . . . . . . . . .  1   1. mm/dd/yy  2. dd/mm/yy
                                                     3. yy/mm/dd  4. ddd/yy

    Execution environment . . . . . . . . . . .  1   1. Online    2. Batch

    Default execution mode. . . . . . . . . . .  1   1. Step      2. Fast

    Default print destination . . . . . . . . .

    Default print class . . . . . . . . . . . .     1



Enter  F1=Help  F3=Exit  F4=Prev  F5=Next  F8=Fwd
```

After you specify basic information about the Department application, you can define the application's responses and functions as described below. Pressing [PF5] from the General Options screen will bring you to the Response/Function List screen.

7.3.4 Step 4: Define application response and function relationships

The **Response/Function List** screen is the main ADSA screen. You can reach this screen by pressing [PF5] from the General Options screen or by entering **2** at the **Screen** prompt on the Main Menu.

Sample screen



```
                              Response/Function List           Page  1 of  1

 Application name:  XXXAPPL    Version:    1

 Select     Response    Assigned      Select    Function              Program/
  (/)       name        key            (/)      name/type(1,2,3)*     Dialog name

   _        ________    _____           _       ________ / _          ________

   _        ________    _____           _       ________ / _          ________

   _        ________    _____           _       ________ / _          ________

   _        ________    _____           _       ________ / _          ________

   _        ________    _____           _       ________ / _          ________

   _        ________    _____           _       ________ / _          ________
                                             * Type: 1. Dialog  2. Program  3. Menu


 Enter  F1=Help  F3=Exit  F4=Prev  F5=Next  F7=Bkwd  F8=Fwd
```

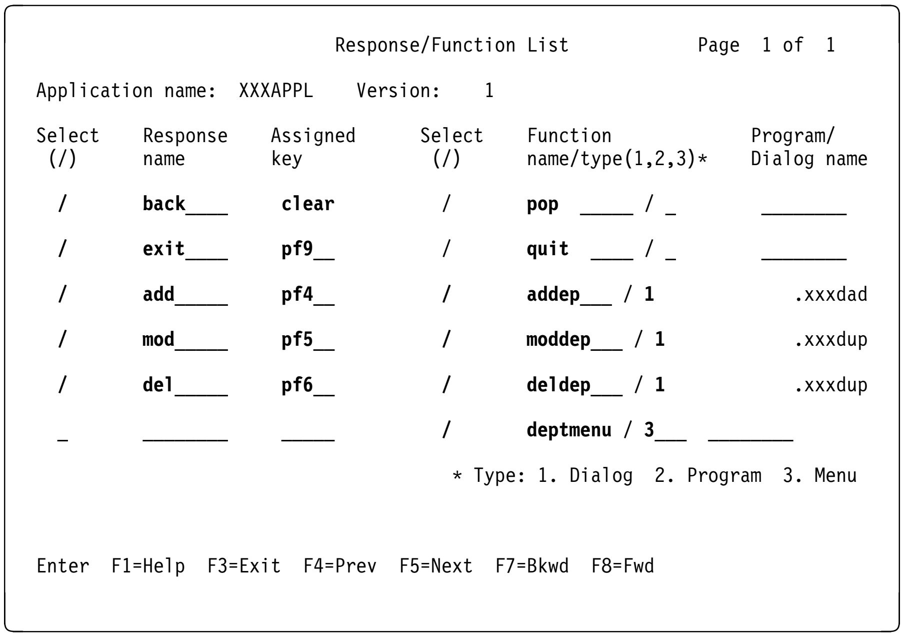
Each response in an application defines a path from one function to another. You are going to name the responses and functions and identify the key associated with each response and the dialog (if any) associated with each function as well as the function type.

Department application responses and functions: Below are the responses and functions invoked in the sample Department application. Each response is associated with a control key and invokes a function. A function can be associated with a dialog.

Response name	**Key**	**Function name**	**Function type**	**Dialog name**	
BACK	CLEAR	POP			Takes the user from ADDDEP, MODDEP, or DELDEP back to the DEPTMENU function.
EXIT	PF9	QUIT			Exits the user from the appli- cation; EXIT is avail- able from any function in the Depart- ment appli- cation
ADD	PF4	ADDDEP	Dialog	*XXX*DADD	Takes the user from the DEPTMENU function to the ADDDEP function, which allows the use to add a new depart- ment record.
MOD	PF5	MODDEP	Dialog	*XXX*DUPD	Takes the user from DEPTMENU to the MODDEP function, which allows the user to modify an existing depart- ment record.
DEL	PF6	DELDEP	Dialog	*XXX*DUPD	Takes the user from DEPTMENU to the DELDEP function, which allows the user to delete an existing depart- ment record.
		DEPTMENU	Menu		Provides a menu

Entering Department application responses and functions: You name each response and function on the **Response/Function List** screen as shown below. Responses and dialog, program, and menu functions need further definition on subsequent screens. Select each response and function requiring further definition by entering a nonblank (nonunderscore) character under **Select** as you go along.

Note: **System** functions, such as **pop** and **quit** are defined by CA-ADS and do not require further definition on the part of the developer.



```
                              Response/Function List          Page  1 of  1

Application name:  XXXAPPL    Version:    1

Select     Response    Assigned      Select     Function             Program/
 (/)       name        key            (/)       name/type(1,2,3)*    Dialog name

  /        back____     clear          /        pop  _____ / _        ________

  /        exit____     pf9__          /        quit  ____ / _        ________

  /        add_____     pf4__          /        addep___ / 1              .xxxdad

  /        mod_____     pf5__          /        moddep___ / 1             .xxxdup

  /        del_____     pf6__          /        deldep___ / 1             .xxxdup

  _        ________     _____          /        deptmenu / 3___  _______

                                               * Type: 1. Dialog  2. Program  3. Menu


Enter  F1=Help  F3=Exit  F4=Prev  F5=Next  F7=Bkwd  F8=Fwd
```

Responses: For each response, enter the response name and name the assigned key.

Functions: For each function, enter the function name and specify the type (**1 - dialog; 2 - program; 3 - menu**) and the dialog associated with it, if there is one.

Note: The dialog names can be specified early in the prototyping process; the dialogs need not exist.

Unique identifier for the response: The combination of the name, key, and associated function for a response makes up the unique identifier for the response within the application definition. Because of this, you can have several different responses with the same name and/or control key. For example, several ADD responses can be defined for an application. One ADD response can invoke function ADDDEP. Another ADD response can invoke function ADDEMP. This capability allows you to define a consistent user interface for an application, where all similar functions are invoked in similar ways.

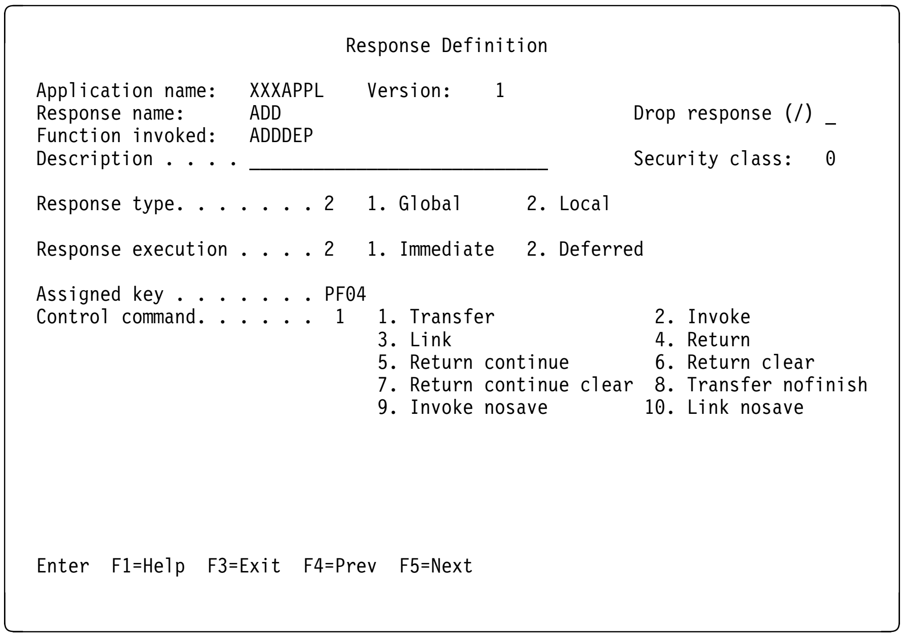
Once you have named all responses and functions, you are ready to further define each one. Pressing [PF5] will advance you through the responses and functions selected on the Response/Function List screen.

Below is a discussion of defining responses and functions.

7.3.5 Step 5: Further define the application responses

You further define each application response by using the **Response Definition** screen. A sample Response Definition screen is shown below:

Sample screen



```
                              Response Definition

Application name:   XXXAPPL    Version:    1
Response name:      ADD                                 Drop response (/) _
Function invoked:   ADDDEP
Description . . . . ____________________________        Security class:   0

Response type. . . . . . . 2   1. Global      2. Local

Response execution . . . . 2   1. Immediate   2. Deferred

Assigned key . . . . . . . PF04
Control command. . . . . .  1   1. Transfer              2. Invoke
                                3. Link                  4. Return
                                5. Return continue       6. Return clear
                                7. Return continue clear  8. Transfer nofinish
                                9. Invoke nosave        10. Link nosave




Enter  F1=Help  F3=Exit  F4=Prev  F5=Next
```

Screen prompts: Certain information will be presented to you when this screen is displayed:

- **Application name** — The name of the application will be displayed.
- **Version** — The version number of the application will be displayed.
- **Response name** — The name of the response as listed on the Response/Function List screen will be displayed. At runtime, the user can enter the response name on the screen to select the response.
- **Function invoked** — The name of the function as listed on the Response/function List screen will be displayed. This is the function that the response invokes at runtime. For example, the ADD response invokes the ADDDEP function at runtime.
- **Assigned key** — The key assigned to this response on the Response/Function List screen will be displayed. At runtime, the user can press the specified control key to select the response.

When you define a response on the Response Definition screen, you typically enter information after the following prompts:

- **Description** — You can type a one-line description for each response. At runtime, the description is displayed on each menu from which the response is available.
- **Security class** — You can assign a security class to this response. Applicable to online applications only, security class specifies the DC/UCF security class in the range 1 to 256, assigned to the application. Zero (0) is defined as always unsecured. See your security administrator about the security class conventions being used at your site.

  >> For more information on security classes, refer to the *CA-ADS Reference*.

- **Response type** — (Applies at definition time only) The default response type, **2 (local)**, specifies that you must explicitly make the response valid for any functions that use the response. You assign this response type to responses that are not used by many functions in the application. For example, it is easier to define ADD as a local response because you add it to only one function definition (DEPTMENU).

  A response that is available from many application functions is typically defined as a **1 (global)** response. A global response is added to each function definition as the function is defined. You later deselect the global response from the few functions that do not use the response. For example, you define BACK as a global response because it is used by all but one function in the application. When defining functions, it is easier to deselect BACK from one function than to explicitly select it for all other functions.

  Note: It is often useful to list the global responses before local responses on the Response/Function List screen so that they are available for all function definitions.

- **Response execution** — You can specify whether the invoked function is immediately executable or deferred.

  >> For more information on immediate and deferred execution, see *CA-ADS Reference*.

- **Control command** — You must specify the control command (for example, **Transfer** or **Link**) the response uses to invoke the specified function at runtime. For prototype applications, which don't perform any significant processing, you typically use the default control command, **1 (Transfer)**. When this command is used to pass control to another function, the runtime system frees resources being held for the original function.

  >> For more information on control commands, see *CA-ADS Reference*.

Responses for Department application: The table below summarizes the specifications you will make for each of the five responses in the Department application. You can define these responses in any order, although it is often useful to define the global responses first. In the sample definition session shown in this chapter, these responses are defined in the order presented below.

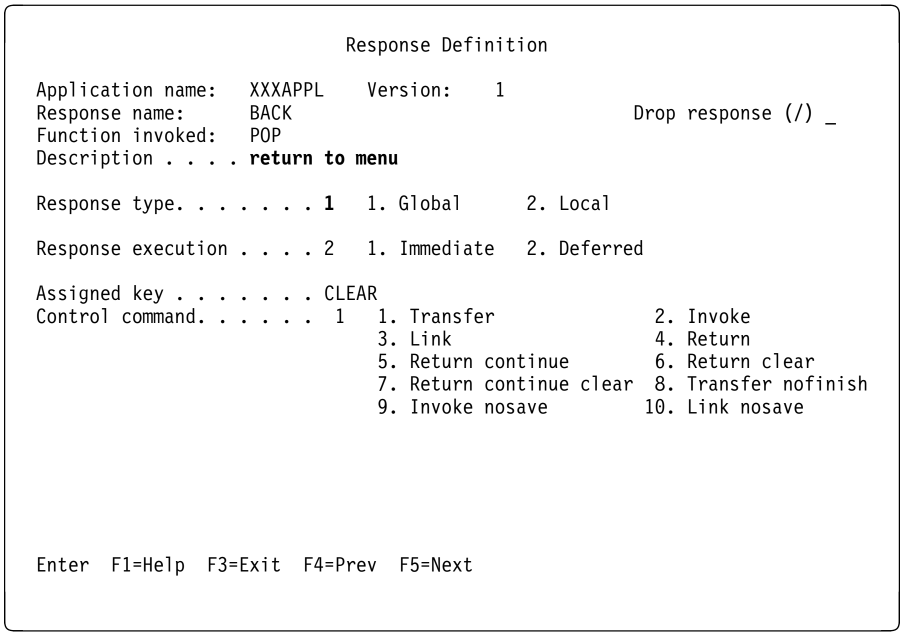
Response name	**Description**	**Control key**	**Function invoked**	**Response type**
BACK	Return to menu	CLEAR	POP	Global
EXIT	Terminate application	PF9	QUIT	Global
ADD	Add a new department	PF4	ADDDEP	Local
MOD	Modify a department	PF5	MODDEP	Local
DEL	Delete a department	PF6	DELDEP	Local

To begin defining application responses, access the Response Definition screen by pressing [PF5] from the Response/Function List screen and define an application response.

The first two responses, BACK and EXIT, are used by most functions in the application. To make the functions easier to define later, you will define BACK and EXIT as **global responses**. When you define functions in ADSA, a global response is automatically available from each function. You can selectively remove a global function from functions when you define the functions.

You define the BACK response as shown:

Defining the BACK response


```
                               Response Definition

Application name:   XXXAPPL    Version:    1
Response name:      BACK                                 Drop response (/) _
Function invoked:   POP
Description . . . . return to menu

Response type. . . . . . . 1   1. Global      2. Local

Response execution . . . . 2   1. Immediate   2. Deferred

Assigned key . . . . . . . CLEAR
Control command. . . . . .  1   1. Transfer              2. Invoke
                                3. Link                  4. Return
                                5. Return continue       6. Return clear
                                7. Return continue clear  8. Transfer nofinish
                                9. Invoke nosave        10. Link nosave





Enter  F1=Help  F3=Exit  F4=Prev  F5=Next
```

After you press [Enter], ADSA adds the response definition if there are no errors, and then redisplays the Response Definition screen.

Defaults: ADSA fills in default values for the following Response Definition screen prompts if you have not entered another value:

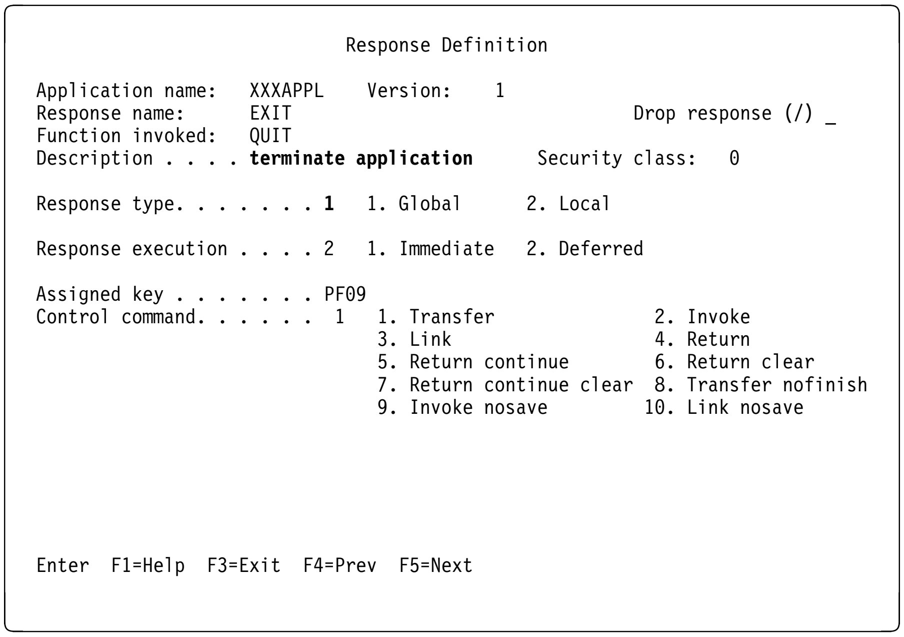
- **Response type** returns the default value of **2 (Local)**. BACK is defined as a **global** response because it is available from many functions in the Department application. To specify that this response is global, overtype 2 with **1**.
- **Response execution** returns the default value of **2 (Deferred)**
- **Control command** returns the default value of **1 (Transfer)**.

When you press [Enter], the screen will redisplay with a confirming message if the definition is correct, or an error message if an error has been encountered.

Note: If you press [Enter] after providing information, the screen will redisplay. If you press [PF5] after providing information **and there are no errors**, the appropriate Function Definition screen will be displayed. Since BACK and EXIT are responses that invoke **system** functions, the Function Definition screen will not be displayed.

Press [PF5] and define the EXIT response in the same way:

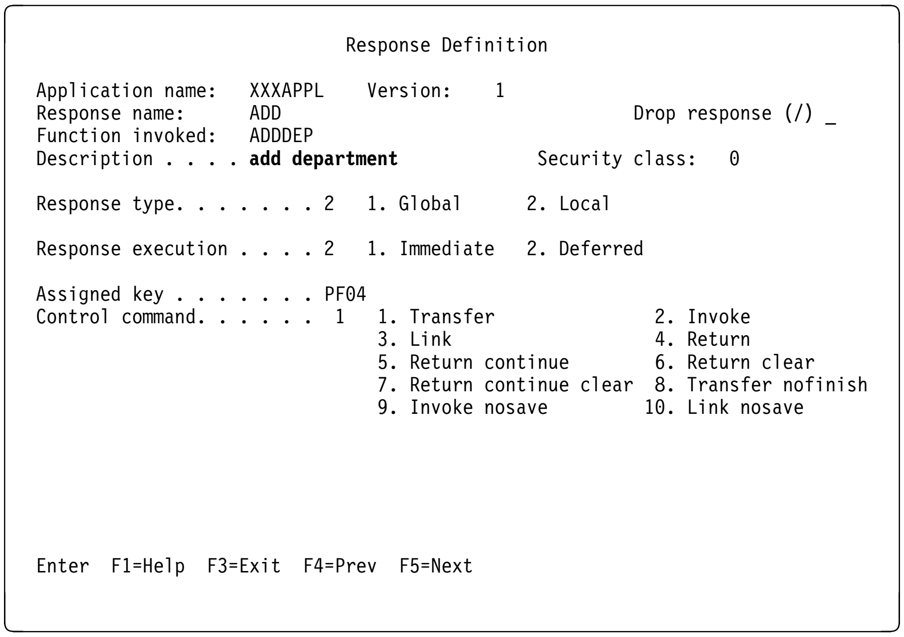
Defining the EXIT response



```
                              Response Definition

Application name:   XXXAPPL    Version:    1
Response name:      EXIT                               Drop response (/) _
Function invoked:   QUIT
Description . . . . terminate application      Security class:   0

Response type. . . . . . . 1   1. Global      2. Local

Response execution . . . . 2   1. Immediate   2. Deferred

Assigned key . . . . . . . PF09
Control command. . . . . .  1   1. Transfer              2. Invoke
                                3. Link                  4. Return
                                5. Return continue       6. Return clear
                                7. Return continue clear  8. Transfer nofinish
                                9. Invoke nosave        10. Link nosave





Enter  F1=Help  F3=Exit  F4=Prev  F5=Next
```

These responses invoke **system** functions. You do not have to further define any system function.

Defining the ADD response: The ADD response invokes the function, ADDDEP. The response definition is as follows:



```
                              Response Definition

Application name:   XXXAPPL    Version:    1
Response name:      ADD                                Drop response (/) _
Function invoked:   ADDDEP
Description . . . . add department            Security class:   0

Response type. . . . . . . 2   1. Global      2. Local

Response execution . . . . 2   1. Immediate   2. Deferred

Assigned key . . . . . . . PF04
Control command. . . . . .  1   1. Transfer              2. Invoke
                                3. Link                  4. Return
                                5. Return continue       6. Return clear
                                7. Return continue clear  8. Transfer nofinish
                                9. Invoke nosave        10. Link nosave




Enter  F1=Help  F3=Exit  F4=Prev  F5=Next
```

ADD is defined as a local response because it is available only from the DEPTMENU function in the Department application. You will make ADD valid for DEPTMENU when you define the DEPTMENU function in Step 6.

After you press [Enter], ADSA adds the response definition if there are no errors, and then redisplays the Response Definition screen.

When you are finished defining a response for the Department application, you define the function it invokes as described below. Pressing [PF5] will display the Function Definition screen.

7.3.6 Step 6: Further define the application functions

Each function in an application represents a unit of work. For example, the DEPTMENU function lists available responses and allows the user to select a response. In this step, you will further define the following Department application functions:

- The **DEPTMENU menu function**, which displays a list of available responses (ADD, MOD, DEL, and EXIT)
- The **ADDDEP dialog function**, which allows the user to add a department record
- The **MODDEP dialog function**, which allows the user to modify an existing department record
- The **DELDEP dialog function**, which allows the user to delete an existing department record

Note: You do not have to define the POP or QUIT functions. POP and QUIT are system functions that have reserved names and meanings. ADSA automatically will add complete definitions for these functions when you associate them with the BACK and EXIT responses.

Function Definition screens: You define menu and dialog functions in the Department application by using one of several Function Definition screens:

- **Function Definition (Dialog)** screen allows you to define dialog functions
- **Function Definition (Menu)** screen allows you to define menu functions
- **Function Definition (Program)** screen allows you to define program functions

>> For further information on program functions, see *CA-ADS Reference*.

Some of these screens are made up of multiple pages accessed through [PF7] and [PF8]. The appropriate screen will be displayed based on the initial function definition you gave on the Response/Function List screen.

Screen prompts: Certain information will be presented to you when this screen is displayed:

- **Application name** — The name of the application will be displayed.
- **Version** — The version number of the application will be displayed.
- **Function name** — The name of the function as listed on the Response/Function List screen will be displayed. ADDDEP, for each function.
- **Associated dialog** (dialog functions only) — The name of the dialog associated with this function will be displayed. A dialog is an application component that typically displays a screen to the user and processes information.

Note: The dialogs that you name in this chapter are not yet defined; you will define dialogs for the Department application in Chapter 9, “Defining Dialogs Using ADSC” on page 9-1.

Use the **Function Definition** screen to define basic information about a function, such as:

- **A description** — You can specify a one-line description to help you identify the purpose of each function. This description is also available to the programmer (to place as a header on a map, for example) through the global record.
- The **responses** that the user can select from the function.

  For example, as shown earlier in this chapter, ADD, MOD, DEL, and EXIT are valid from the DEPTMENU function. EXIT is a global response, so it is automatically available from DEPTMENU. You will use the Function Definition screen to make ADD, MOD, and DEL also available from DEPTMENU.
- Specify how menu screens are to appear at runtime (resequencing of menu items and addition of header information). CA-ADS uses your specifications to format and display menus at runtime.

>> For more information on these and other ADSA screens, see *CA-ADS Reference*.

The Department application: The following table summarizes the specifications you will make for the functions in the Department application. You define one application function at a time, transferring between ADSA screens as necessary. Instructions for defining the DEPTMENU menu function and for defining the ADDDEP, MODDEP, and DELDEP dialog functions are presented below.

Function name	Description	Associated dialog	Function type	Valid responses
ADDDEP	Add department	*XXX*DADD	1 (Dialog)	BACK EXIT
MODDEP	Mod department	*XXX*DUPD	1 (Dialog)	BACK EXIT
DELDEP	Del department	*XXX*DUPD	1 (Dialog)	BACK EXIT
DEPTMENU	Department menu	Not applicable	3 (Menu)	ADD MOD DEL EXIT

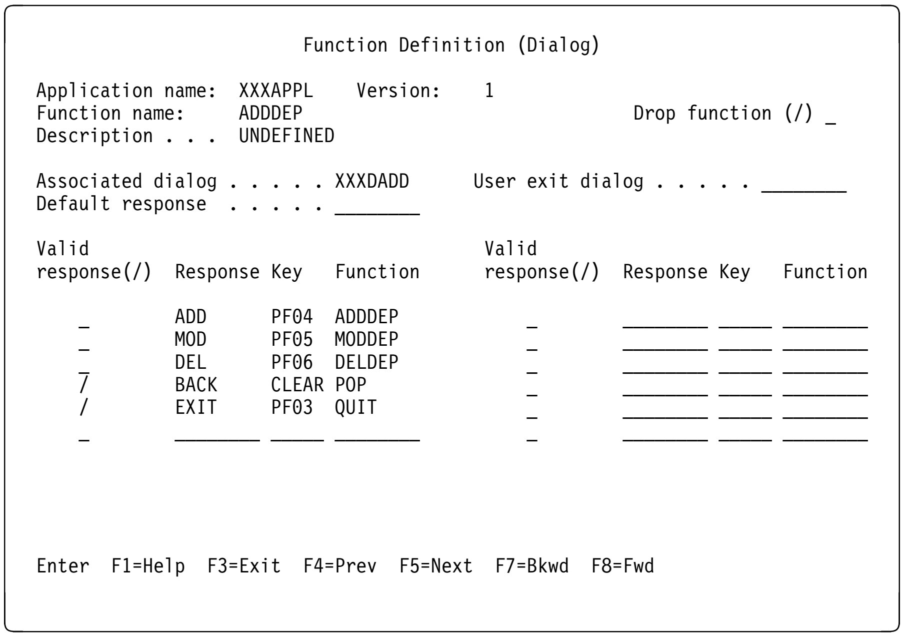
Dialog and menu functions are discussed separately below. Remember, however, that these screens will be displayed in the order in which you selected the functions on the Response/Function List screen. Pressing [PF5] will move you from a response definition to a function definition and then on to the next response definition.

7.3.6.1 Dialog functions

The functions, ADDDEP, MODDEP, and DELDEP, are all dialog functions. You named these dialog functions on the Response/Function List screen earlier in this chapter. At that time, ADSA automatically added *skeleton* definitions for the associated ADDDEP, MODDEP, and DELDEP functions. When you further define a dialog function, you enhance these skeleton function definitions.

Pressing [PF5] from the Response Definition screen for the response ADD brings you to the dialog function definition for dialog ADDDEP in the Department application.

Sample screen



```
                          Function Definition (Dialog)

Application name:  XXXAPPL    Version:    1
Function name:     ADDDEP                           Drop function (/) _
Description . . .  UNDEFINED

Associated dialog . . . . . XXXDADD      User exit dialog . . . . . ________
Default response  . . . . . ________

Valid                                     Valid
response(/)  Response Key   Function      response(/)  Response Key   Function

    _        ADD      PF04  ADDDEP            _        ________ _____ ________
    _        MOD      PF05  MODDEP            _        ________ _____ ________
    _        DEL      PF06  DELDEP            _        ________ _____ ________
    /        BACK     CLEAR POP               _        ________ _____ ________
    /        EXIT     PF03  QUIT              _        ________ _____ ________
    _        ________ _____ ________          _        ________ _____ ________




Enter  F1=Help  F3=Exit  F4=Prev  F5=Next  F7=Bkwd  F8=Fwd
```

You use the **Function Definition (Dialog)** screen to specify basic information for the dialog function, including a description, and valid responses for that function.

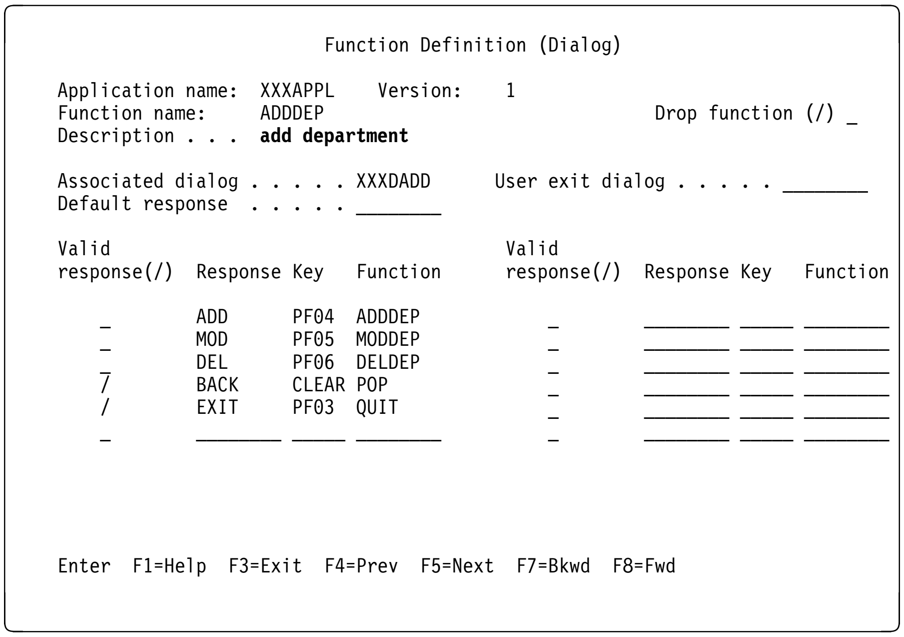
Function description: When you display the Function Definition screen, the **Description** field will contain a value of **UNDEFINED** because you have not yet provided a description. A description must be provided before the application can be compiled.

Valid responses: Valid responses are the responses (for example, BACK or EXIT) that a user can access directly from a function. You specify a valid response by entering a nonblank character opposite the response. When the screen is refreshed, a slash (/) is displayed.

A response that has been defined as **global** on the Response Definition screen will already be selected as a valid response for this function. You can deselect a response by spacing over the slash.

To make it easier for you to define dialog functions, dialogs that you name on the Function Definition screen do not yet have to be defined in the data dictionary. You will define dialogs for Department application dialog functions later in this manual, in Chapter 9, "Defining Dialogs Using ADSC" on page 9-1.

Defining the ADDDEP function: You specify basic information for the ADDDEP function as shown:



```
                        Function Definition (Dialog)

Application name:  XXXAPPL    Version:    1
Function name:     ADDDEP                                  Drop function (/) _
Description . . .  add department

Associated dialog . . . . . XXXDADD      User exit dialog . . . . . ________
Default response  . . . . . ________

Valid                                     Valid
response(/)  Response Key   Function      response(/)  Response Key   Function

    _        ADD      PF04  ADDDEP            _        ________ _____ ________
    _        MOD      PF05  MODDEP            _        ________ _____ ________
    _        DEL      PF06  DELDEP            _        ________ _____ ________
    /        BACK     CLEAR POP               _        ________ _____ ________
    /        EXIT     PF03  QUIT              _        ________ _____ ________
    _        ________ _____ ________          _        ________ _____ ________




Enter  F1=Help  F3=Exit  F4=Prev  F5=Next  F7=Bkwd  F8=Fwd
```

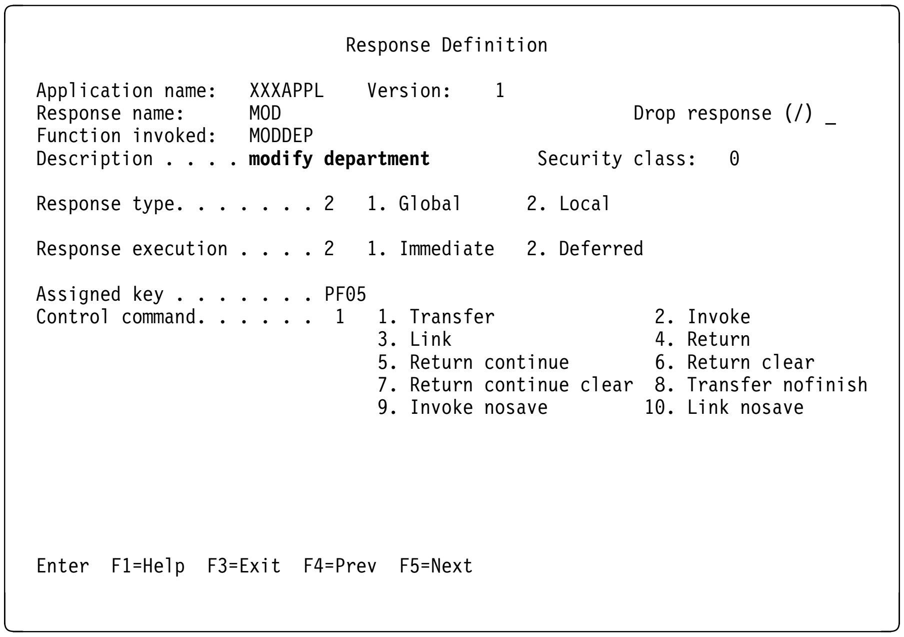
When you press [Enter], ADSA redisplays the Function Definition screen with an appropriate message. A confirming message is returned if there are no errors. In this case, you have successfully modified and enhanced the skeleton dialog function.

An error message is returned if ADSA detects any errors. In this case, use the message to determine the problem. You can type over any errors, and then press [Enter] again.

Press [PF5] to see the next Response Definition screen.

Note: If you press [PF5] rather than [Enter] after providing information **and there are no errors**, the appropriate Response Definition screen (or Function Definition screen, if there are no more responses to be defined) is displayed immediately.

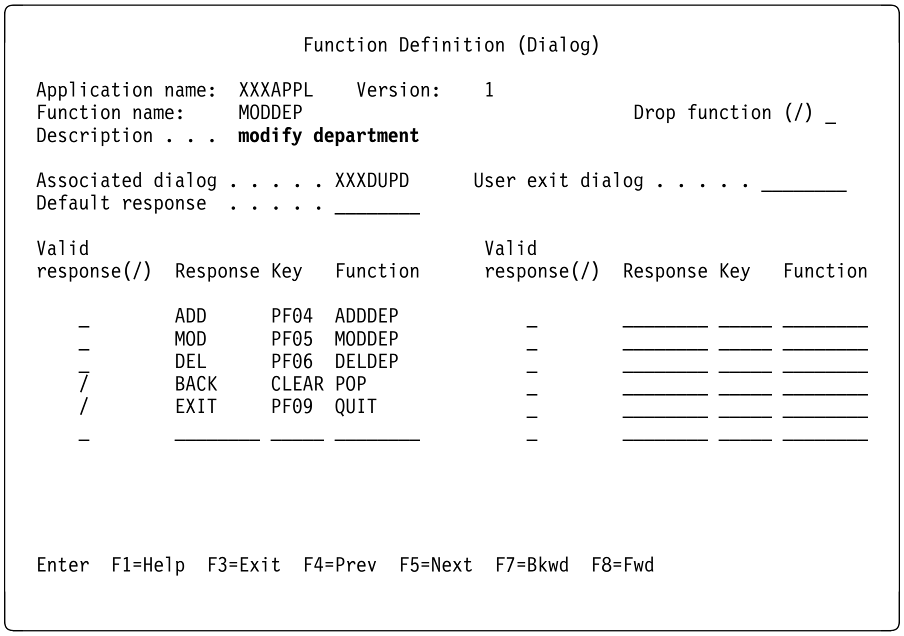
Defining the MOD response: You can now define the MOD response.



```
                             Response Definition

Application name:   XXXAPPL    Version:    1
Response name:      MOD                                  Drop response (/) _
Function invoked:   MODDEP
Description . . . . modify department         Security class:   0

Response type. . . . . . . 2   1. Global      2. Local

Response execution . . . . 2   1. Immediate   2. Deferred

Assigned key . . . . . . . PF05
Control command. . . . . .  1   1. Transfer              2. Invoke
                                3. Link                  4. Return
                                5. Return continue       6. Return clear
                                7. Return continue clear  8. Transfer nofinish
                                9. Invoke nosave        10. Link nosave





Enter  F1=Help  F3=Exit  F4=Prev  F5=Next
```

Defining the MODDEP function: Press [PF5] to see the Function Definition screen for the function that response MOD invokes, MODDEP.

You now can define the **MODDEP** dialog function by using the Function Definition screen:

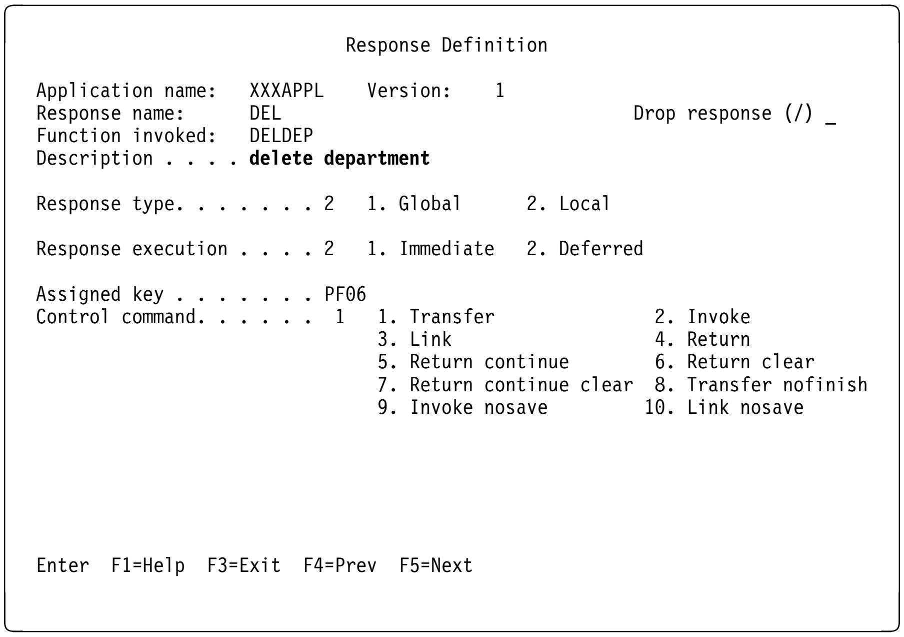


```
                          Function Definition (Dialog)

Application name:  XXXAPPL    Version:    1
Function name:     MODDEP                                Drop function (/) _
Description . . .  modify department

Associated dialog . . . . . XXXDUPD      User exit dialog . . . . . ________
Default response  . . . . . ________

Valid                                     Valid
response(/)  Response Key   Function      response(/)  Response Key   Function

    _        ADD      PF04  ADDDEP             _       ________ _____ ________
    _        MOD      PF05  MODDEP             _       ________ _____ ________
    _        DEL      PF06  DELDEP             _       ________ _____ ________
    /        BACK     CLEAR POP                _       ________ _____ ________
    /        EXIT     PF09  QUIT               _       ________ _____ ________
    _        ________ _____ ________           _       ________ _____ ________




Enter  F1=Help  F3=Exit  F4=Prev  F5=Next  F7=Bkwd  F8=Fwd
```

When you are finished defining MODDEP, press [PF5] to go directly to the next selected response.

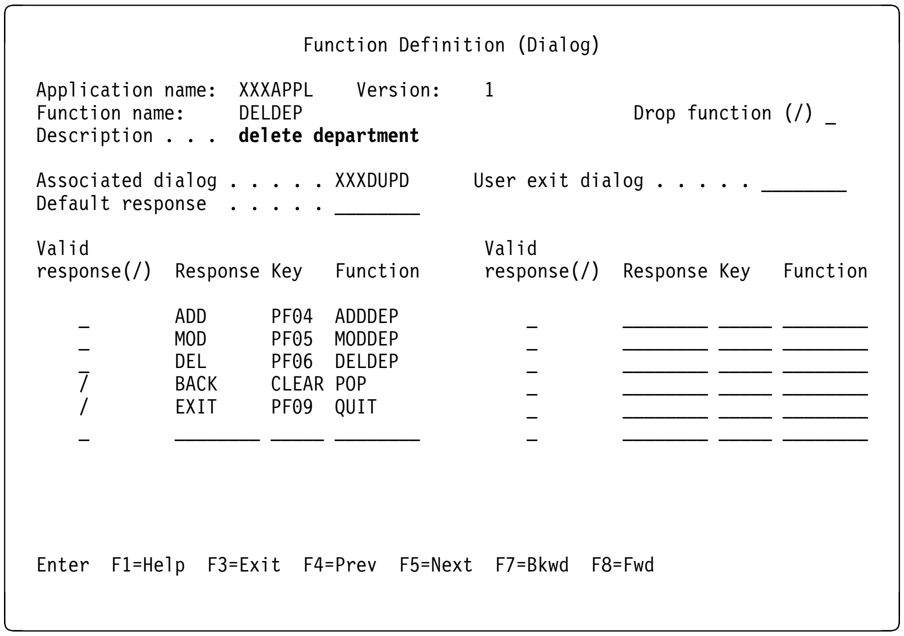
Use the Response Definition and Function Definition screens to define the DEL response and the DELEMP function.

Defining the DEL response



```
                              Response Definition

Application name:   XXXAPPL    Version:    1
Response name:      DEL                                  Drop response (/) _
Function invoked:   DELDEP
Description . . . . delete department

Response type. . . . . . . 2   1. Global      2. Local

Response execution . . . . 2   1. Immediate   2. Deferred

Assigned key . . . . . . . PF06
Control command. . . . . .  1   1. Transfer              2. Invoke
                                3. Link                  4. Return
                                5. Return continue       6. Return clear
                                7. Return continue clear  8. Transfer nofinish
                                9. Invoke nosave        10. Link nosave




Enter  F1=Help  F3=Exit  F4=Prev  F5=Next
```

Defining the DELDEP function



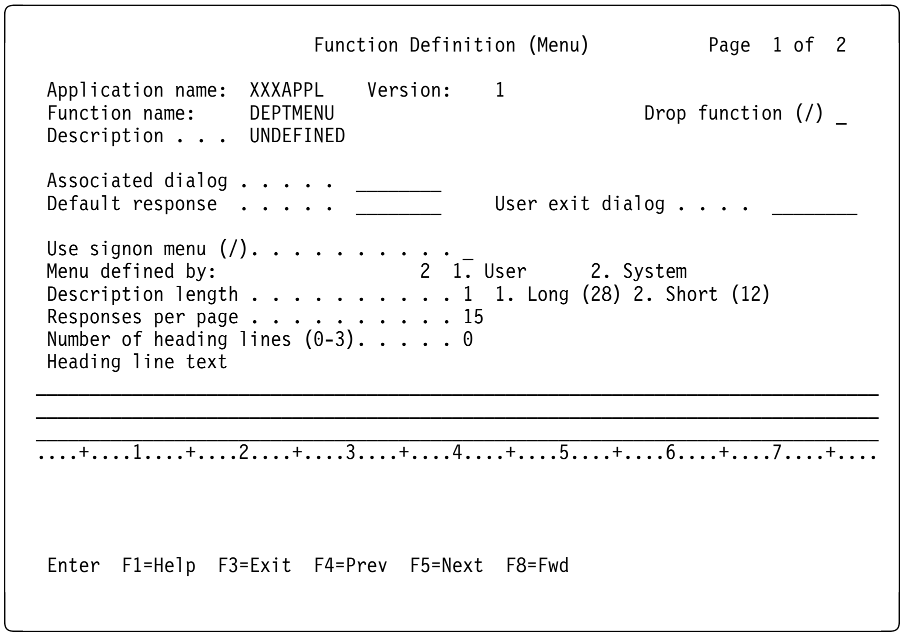
```
                          Function Definition (Dialog)

Application name:  XXXAPPL    Version:    1
Function name:     DELDEP                                 Drop function (/) _
Description . . .  delete department

Associated dialog . . . . . XXXDUPD     User exit dialog . . . . . ________
Default response  . . . . . ________

Valid                                   Valid
response(/)  Response Key   Function     response(/)  Response Key   Function

    _        ADD      PF04  ADDDEP           _        ________ _____ ________
    _        MOD      PF05  MODDEP           _        ________ _____ ________
    _        DEL      PF06  DELDEP           _        ________ _____ ________
    /        BACK     CLEAR POP              _        ________ _____ ________
    /        EXIT     PF09  QUIT             _        ________ _____ ________
    _        ________ _____ ________         _        ________ _____ ________




Enter  F1=Help  F3=Exit  F4=Prev  F5=Next  F7=Bkwd  F8=Fwd
```

Once you have defined the responses and functions above, press PF5 to bring you to another Function Definition screen. There is one function as yet undefined: DEPTMENU. DEPTMENU is a **menu function**.

7.3.6.2 Menu functions

The last function listed in the table above is DEPTMENU, a menu function. You specified that it was a menu function on the Response/Function List screen in Step 3. To further define DEPTMENU, you must access the Function Definition screen by pressing [PF5] from the Response Definition screen.

Sample screen



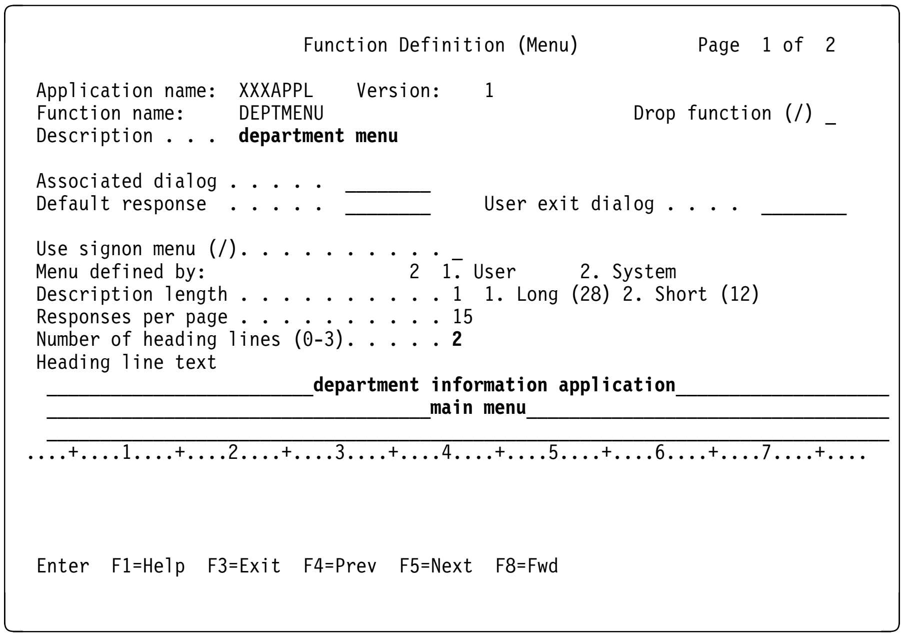
```
                           Function Definition (Menu)          Page  1 of  2

  Application name:  XXXAPPL    Version:    1
  Function name:     DEPTMENU                              Drop function (/) _
  Description . . .  UNDEFINED

  Associated dialog . . . . .  ________
  Default response  . . . . .  ________     User exit dialog . . . .  ________

  Use signon menu (/). . . . . . . . . . _
  Menu defined by:                  2  1. User      2. System
  Description length . . . . . . . . . . 1  1. Long (28) 2. Short (12)
  Responses per page . . . . . . . . . . 15
  Number of heading lines (0-3). . . . . 0
  Heading line text
________________________________________________________________________________
________________________________________________________________________________
________________________________________________________________________________
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....



  Enter  F1=Help  F3=Exit  F4=Prev  F5=Next  F8=Fwd
```

You use the **Function Definition (Menu)** screen to specify basic information about a menu function. Each Function Definition (Menu) screen is made up of two pages. You can access the second page of the screen by pressing [PF8].

Menu screens: Menu screens are supplied by CA-ADS; you do not have to write any statements to handle the display or operation of a menu. To tailor the appearance of a menu display, you can define a header to be displayed at the top of the menu screen. This header can include a title, instructions, or any other appropriate text.

You specify basic information for function DEPTMENU as shown:

Defining the DEPTMENU function

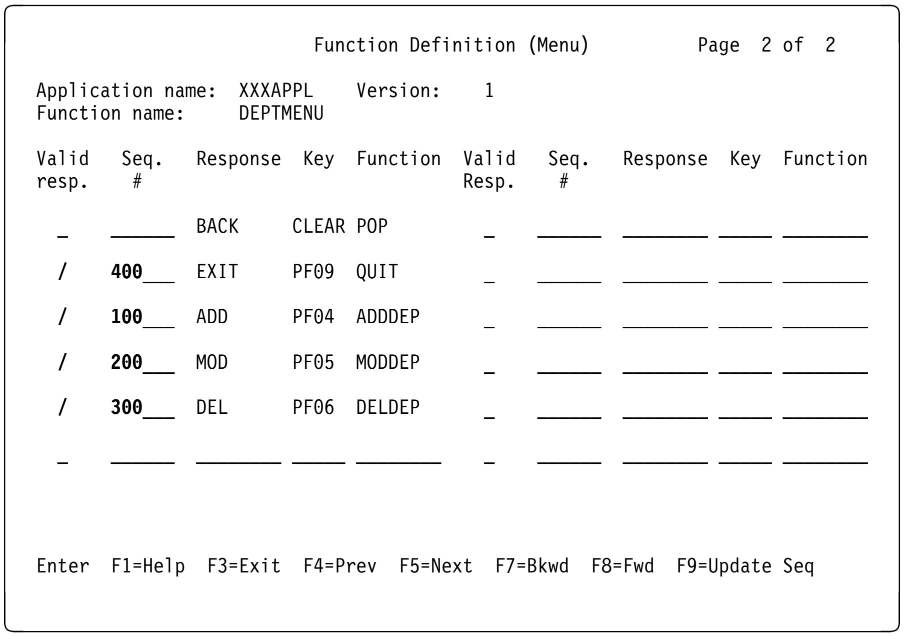


```
                         Function Definition (Menu)          Page  1 of  2

 Application name:  XXXAPPL    Version:    1
 Function name:     DEPTMENU                           Drop function (/) _
 Description . . .  department menu

 Associated dialog . . . . .  ________
 Default response  . . . . .  ________      User exit dialog . . . .  ________

 Use signon menu (/). . . . . . . . . . _
 Menu defined by:                  2  1. User      2. System
 Description length . . . . . . . . . . 1  1. Long (28) 2. Short (12)
 Responses per page . . . . . . . . . . 15
 Number of heading lines (0-3). . . . . 2
 Heading line text
  _________________________department information application___________________
  ____________________________________main menu__________________________________
  _______________________________________________________________________________
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....




 Enter  F1=Help  F3=Exit  F4=Prev  F5=Next  F8=Fwd
```

After you press [Enter], ADSA redisplays the Function Definition screen with a confirming message. An error message is returned if ADSA detects any errors. In this case, use the message to determine the problem. You can type over any errors, and then press [Enter] again.

Second page of Function Definition (Menu): You use the second page of the Function Definition (Menu) screen to specify the responses (such as ADD) that a user can access directly from the function. You also specify the sequence that the response will be displayed on the menu. You access the second page by pressing [PF8]. Second and subsequent pages are response sequence screens.

Access the second page and make the ADD, MOD, DEL, and EXIT responses valid from the DEPTMENU function as shown:



```
                         Function Definition (Menu)        Page  2 of  2

Application name:  XXXAPPL    Version:    1
Function name:     DEPTMENU

Valid   Seq.   Response  Key  Function  Valid   Seq.   Response  Key  Function
resp.    #                              Resp.    #

  _    ______  BACK     CLEAR POP         _    ______  ________ _____ ________

  /    400___  EXIT     PF09  QUIT        _    ______  ________ _____ ________

  /    100___  ADD      PF04  ADDDEP      _    ______  ________ _____ ________

  /    200___  MOD      PF05  MODDEP      _    ______  ________ _____ ________

  /    300___  DEL      PF06  DELDEP      _    ______  ________ _____ ________

  _    ______  ________ _____ ________    _    ______  ________ _____ ________



Enter  F1=Help  F3=Exit  F4=Prev  F5=Next  F7=Bkwd  F8=Fwd  F9=Update Seq
```

Specifying menu sequence: You can use this screen to specify the sequence in which valid responses are displayed on the menu screen.

You do this by entering sequence numbers for each valid response.

If you want to change the sequence numbers, you can change those numbers:

```
100    ADD
200    DEL
300    EXIT          To display MOD between ADD
150    MOD      ◄--- and DEL, you would change
                     400 to some value between
                     100 (ADD) and 200 (DEL).
                     For example: 101 or 150.
```

Response sequence numbers displayed on the Function Definition screen are not displayed to users.

Inhibiting response display: You can also use this screen to inhibit the display of valid responses on the menu screen.

You do this by replacing the sequence number for the response with 0 (zero):

```
100    EMPINFO
200    DEPTINFO      To inhibit display of the
000    SALARIES ◄--- SALARIES response, you
400    EXIT          would replace 300 with
                     0 (zero).
```

Invisible responses can still be accessed by any user who knows the response name or control key. To actively restrict responses, see information on security in the *CA-ADS Reference*.

>> For more information on other uses of the Function Definition (Menu) screen, see *CA-ADS Reference*.

The DEPTMENU function is now fully defined.

After you finish defining functions and responses for the Department application, press [PF5] to return to the Response/Function List screen. Processed selections on the Response/Function List screen will be de-selected; unprocessed selections will still be selected and are accessed when you press [PF5].

It is helpful to re-access the Response/Function List screen to remind yourself where you are in the definition process.

Press [PF5] again to go to the Global Records screen.

Press [PF5] again to go on to the Task Codes screen where you define a task code for the application. (Alternatively, you can press [PF3] from the Response/Function List screen to return to the Main Menu and choose option 4.)

7.3.7 Step 7: Define a task code

A task code defines an entry point into an application. At run time, developers and users can execute an application by using the application's task code. An application can have more than one task code; each one can be associated with a different function in the application.

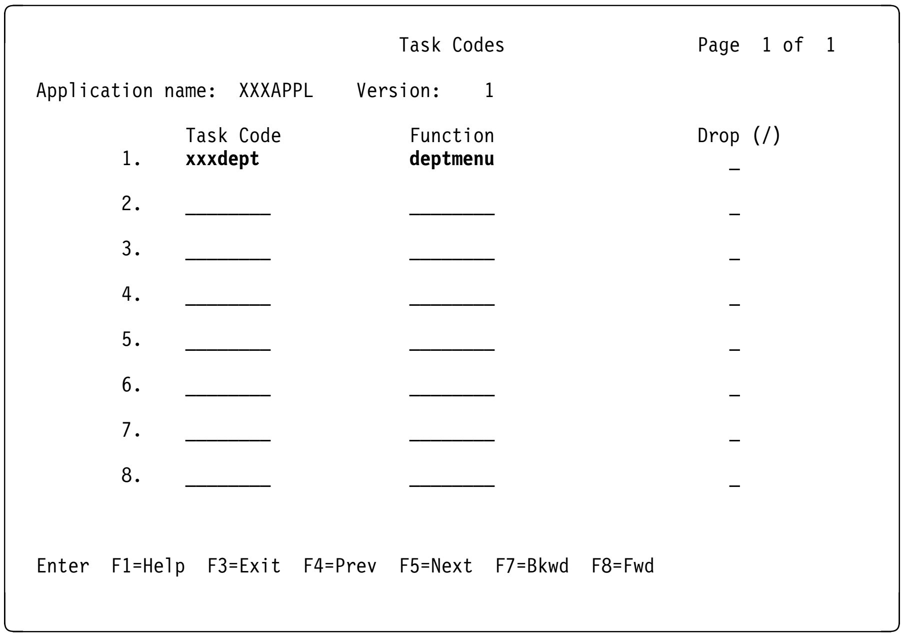
As an application developer, you must define at least one task code for an application before you can successfully create a load module for the application. When you define a task code, you name:

- **A task code** (for example, *XXX*DEPT) that a user can supply to invoke the application
- **The function** (for example, DEPTMENU) to be executed first when a user supplies the associated task code

You use the **Task Codes** screen to define task codes for an application. You can access the Task Codes screen from the Response/Function List screen by pressing [PF5].

Define a task code as shown:

Defining the task code



```
                            Task Codes                Page  1 of  1

Application name:  XXXAPPL    Version:    1

              Task Code            Function                   Drop (/)
        1.    xxxdept              deptmenu                      _

        2.    ________             ________                      _

        3.    ________             ________                      _

        4.    ________             ________                      _

        5.    ________             ________                      _

        6.    ________             ________                      _

        7.    ________             ________                      _

        8.    ________             ________                      _


Enter  F1=Help  F3=Exit  F4=Prev  F5=Next  F7=Bkwd  F8=Fwd
```

After you press [Enter], ADSA redisplays the Task Codes screen with a confirming message. An error message is returned if ADSA detects any errors. In this case, use the message to determine the problem. You can type over errors, and then press [Enter] again.

When you have defined a task code for the Department application, you can create a load module for the application as described below.

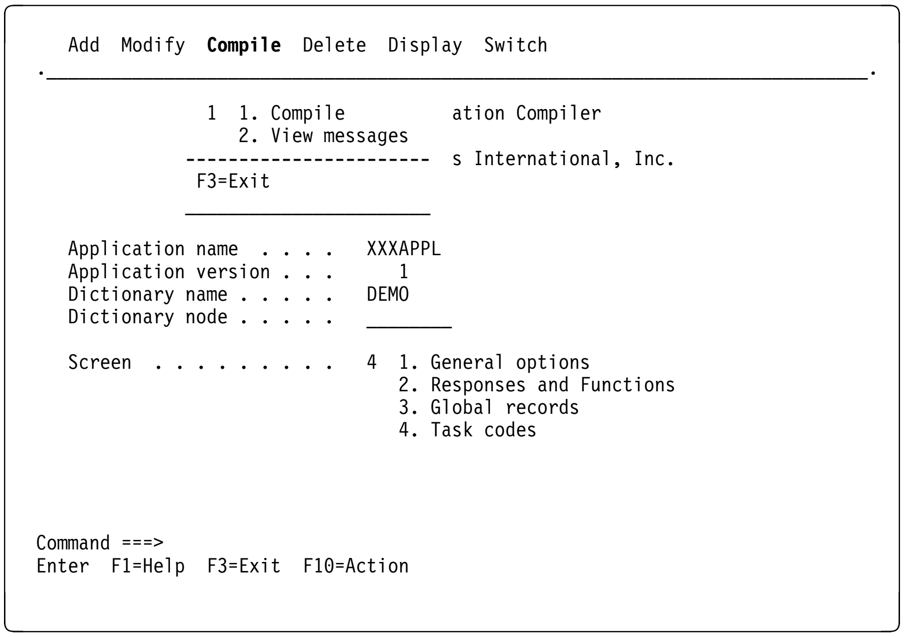
7.3.8 Step 8: Compile the application

When you compile an application, ADSA creates a load module that incorporates all of your specifications. You compile an application by selecting the **Compile** activity from the the action bar on the Main Menu screen.

You compile an application from the Main Menu screen. To get to the Main Menu screen from the Task Codes screen, press [PF5].

Compiling the application: To compile the application, position the cursor on the **Compile** item on the action bar and press ENTER. You can position the cursor on **Compile** by:

- Tabbing to **Compile** and pressing [Enter]
- Pressing [PF10] to move to the action bar and then tabbing to **Compile** and pressing [Enter]
- Typing **compile** on the command line and pressing [Enter]



```
   Add  Modify  Compile  Delete  Display  Switch
 .____________________________________________________________________________.

               1  1. Compile         ation Compiler
                  2. View messages
             -----------------------  s International, Inc.
              F3=Exit
             _______________________

   Application name  . . . .   XXXAPPL
   Application version . . .      1
   Dictionary name . . . . .   DEMO
   Dictionary node . . . . .   ________

   Screen  . . . . . . . . .   4  1. General options
                                  2. Responses and Functions
                                  3. Global records
                                  4. Task codes



Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

Once you have displayed the **Compile** action item, choose **1** and press [Enter] to compile the application.

After you press [Enter] to compile the application, ADSA displays messages to indicate whether the application has been compiled successfully

If you receive an error message you can display diagnostic information by selecting the **View messages** option from the **Compile** activity on the action bar. Based on this information, you can correct the application and then compile it again.

7.3.9 Exit from ADSA

You can return directly to DC/UCF from the Main Menu by pressing PF3. Alternatively, you can use the **Switch** activity on the action bar of the Main Menu screen to transfer to another development tool.

In this sample session, you'll exit to DC/UCF so you can execute your application structure. Press [PF3] to exit.

Note: If you leave ADSA without successfully compiling the current application definition, ADSA saves the suspended definition in a queue record associated with your user ID. In an actual production environment, other users will not be able to access the application definition. To enable them to access the definition, specify the **Release** option from the **Modify** activity on the action bar on the Main Menu.

After you exit from ADSA, you can execute your application as described below.

7.3.10 Optionally execute the application

In the previous steps, you defined an executable application structure by using ADSA screens. At this stage, your menu and system functions are fully defined. Your application prototype is not fully operational, however, until you associate the dialog functions in your application with executable dialogs.

In a typical development environment, application developers don't execute an application until the prototype is complete. If you would like to execute your application to see what you've already created, you can invoke and test the application as described below.

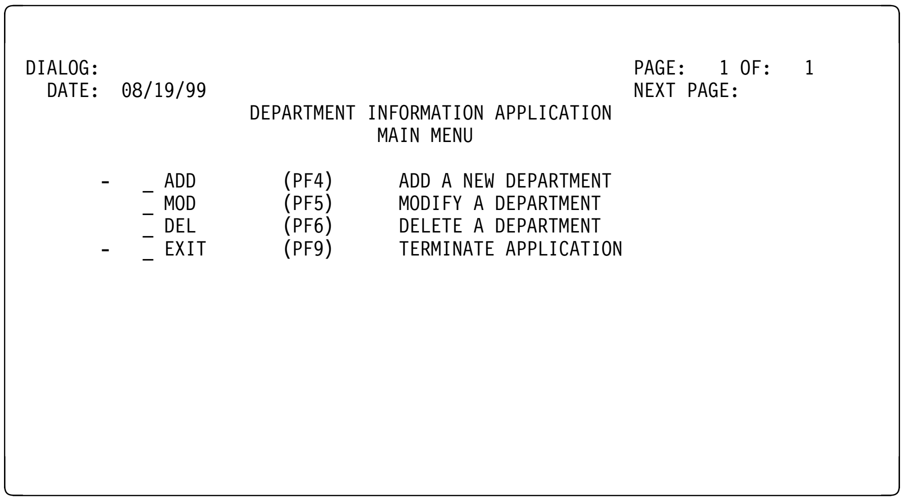
7.3.10.1 Invoke the application

CA-ADS applications execute under the CA-ADS runtime system. To invoke the Department application from DC/UCF, you can enter the task code for the runtime system, followed by the application task code.

For example, assume that ADS is the task code for the runtime system and *XXX*DEPT is the task code for the Department application. You can invoke the Department application as shown:

```
ads xxxdept
 ↑       ↑
 |       |
 |       Sample task code for
 |       the Department application
 |
Sample task code
for the runtime system
                                                   [Enter]
```

The DEPTMENU screen is the first screen displayed.



```
DIALOG:                                           PAGE:   1 OF:   1
  DATE:  08/19/99                                 NEXT PAGE:
                     DEPARTMENT INFORMATION APPLICATION
                                 MAIN MENU

       -   _ ADD        (PF4)      ADD A NEW DEPARTMENT
           _ MOD        (PF5)      MODIFY A DEPARTMENT
           _ DEL        (PF6)      DELETE A DEPARTMENT
       -   _ EXIT       (PF9)      TERMINATE APPLICATION
```

Alternatively, you can enable developers and users to simultaneously invoke the runtime system with an application. To do this, you associate the application task code with the runtime system in either of the following ways:

- **At system runtime** you issue a DCMT VARY DYNAMIC TASK command while using DC/UCF to dynamically associate a task code with the runtime system. Use of the DCMT VARY DYNAMIC TASK command is shown in 9.4, “Instructions for executing the application” on page 9-17, later in this manual.
- **At system-generation time** you use the TASK statement in the system definition to associate the application task code with the CA-ADS runtime system.

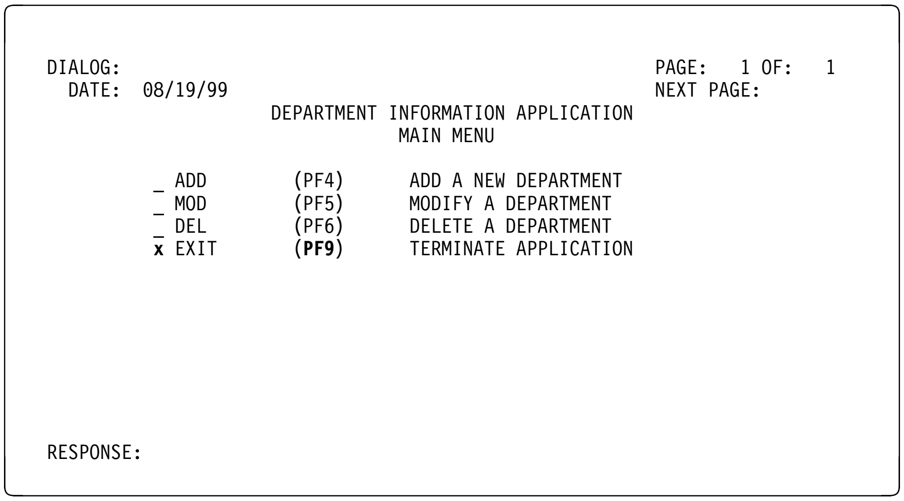
  This procedure typically is used for a production application. For more information, see *CA-IDMS System Generation*.

After you invoke the Department application from DC/UCF, you can test out features that you've already implemented, as described below.

7.3.10.2 Test current features

Using ADSA, you made the ADD, MOD, DEL, and EXIT responses valid from the DEPTMENU function in Step 6. Of these responses, EXIT is the only response that is associated with a fully defined function.

To see how the EXIT response works, select EXIT from the menu screen in any of the following ways:



```
DIALOG:                                              PAGE:   1 OF:   1
  DATE:  08/19/99                                    NEXT PAGE:
                    DEPARTMENT INFORMATION APPLICATION
                               MAIN MENU

          _ ADD        (PF4)      ADD A NEW DEPARTMENT
          _ MOD        (PF5)      MODIFY A DEPARTMENT
          _ DEL        (PF6)      DELETE A DEPARTMENT
          x EXIT       (PF9)      TERMINATE APPLICATION







RESPONSE:
```

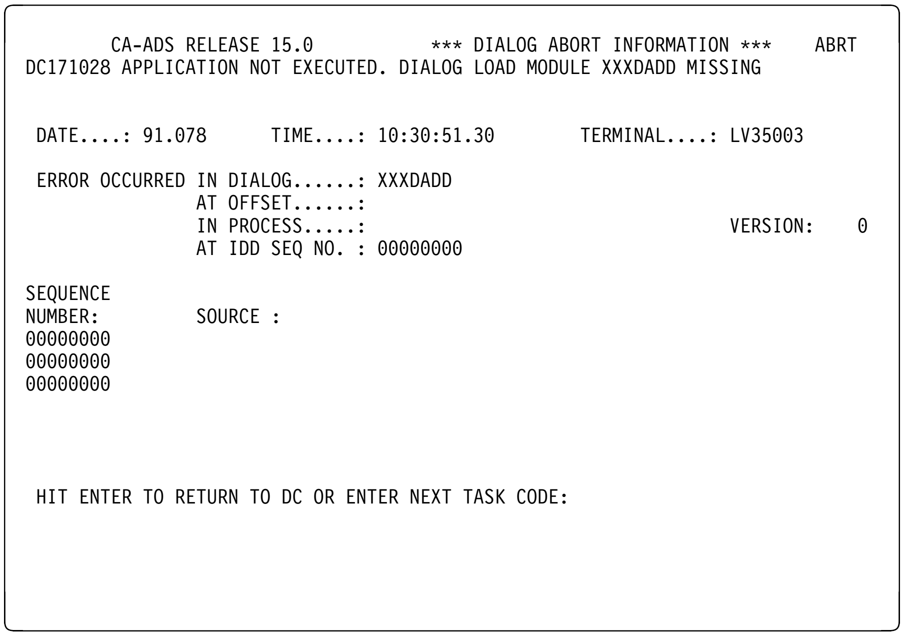
To test out other features of the Department application, you can invoke the application again, as described earlier in 7.3.10.1, “Invoke the application” on page 7-35.

You can try pressing a control key (such as &pf4). that is *not* associated with a response on this menu. The runtime system automatically detects undefined control keys and returns the following message:

```
*** UNACCEPTABLE RESPONSE. PLEASE TRY AGAIN ***
```

Requesting a function that requires further definition: You can try the ADD, MOD, and DEL responses if you want. Since the associated ADDDEP, MODDEP, and DELDEP functions still require further definition, selecting ADD, MOD or DEL will cause the application to terminate. In this case, the runtime system displays the Dialog Abort Information screen:

Dialog Abort Information screen



```
       CA-ADS RELEASE 15.0          *** DIALOG ABORT INFORMATION ***    ABRT
DC171028 APPLICATION NOT EXECUTED. DIALOG LOAD MODULE XXXDADD MISSING


 DATE....: 91.078      TIME....: 10:30:51.30       TERMINAL....: LV35003

 ERROR OCCURRED IN DIALOG......: XXXDADD
                AT OFFSET......:
                IN PROCESS.....:                             VERSION:    0
                AT IDD SEQ NO. : 00000000

SEQUENCE
NUMBER:         SOURCE :
00000000
00000000
00000000




 HIT ENTER TO RETURN TO DC OR ENTER NEXT TASK CODE:
```

The **Dialog Abort Information** screen is particularly useful when you are developing and debugging process logic for dialogs. At that time, this screen can help you determine where in a module of process code the dialog fails.

The display of this diagnostic screen can be disabled when the application is ready for final release.

7.4 Summary

Creating the executable structure: In this chapter, you used ADSA screens to create the executable structure for the sample Department application. The structure includes:

- **Responses**, which define the possible runtime paths available to users of the application
- **Functions**, which define the activities that users can perform while using the application
- **A task code** that defines an entry point into the application and allows users to invoke the application

You built the application structure by establishing relationships between responses and functions as you defined them:

- **For each response**, you named the function to be invoked by the response.
- **For each function**, you named the responses to be valid from the function. At runtime, the user can select any of the valid responses from the function.

For **menu function** DEPTMENU, you also specified how options for the user and the menu's title are to appear on the menu screen at runtime. For each of the application's **dialog functions**, you also named the executable component (that is, the *dialog*) to be executed at runtime when the user invokes the associated function. You will actually define these dialogs in Chapter 9, "Defining Dialogs Using ADSC" on page 9-1.

Creating the load module: When you finished defining the structure of the Department application, you created a **load module** for the application. You created this executable load module without explicitly writing *any* lines of procedural code. By using ADSA screens, you have *implicitly* coded all potential flow of control for the application.

Even at this early stage in the application development cycle, your application contains fully executable components, such as function DEPTMENU. Your dialog functions (ADDDEP, MODDEP, and DELDEP) are not yet developed fully, so you cannot execute them. As soon as you develop the ADDDEP, MODDEP, and DELDEP dialog functions, your application will be fully executable.

The first step in developing dialogs is to create screens that the dialogs will display to users. You will create the screen display for the ADDDEP, MODDEP, and DELDEP dialog functions in the next chapter.

Chapter 8. Defining a Screen Display Using MAPC

8.1 Introduction

In the previous chapter, you used ADSA to define the structure of the sample Department application. As another step in creating an application, you define screen displays by using the map compiler (MAPC). This chapter provides instructions for defining the *XXX*MAP screen display for use in the sample Department application.

Note: When creating your map, you can substitute your initials for the *XXX* in the map name.

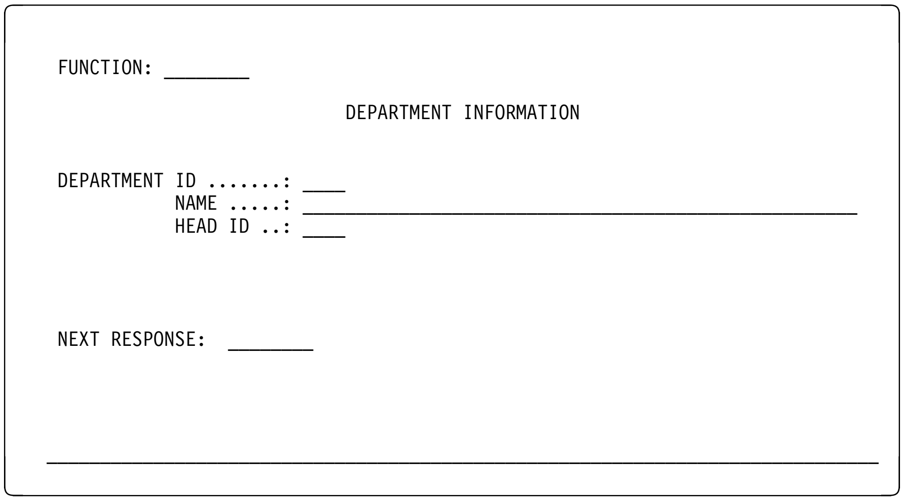
This chapter includes:

- An overview of how maps are used in the CA-ADS environment
- Instructions for defining maps for the sample Department application
- A summary of what you've accomplished in this chapter

8.2 Overview

What is a map: A **.* * map** is a predefined screen display used by dialogs in an application. At runtime, dialogs use maps to interact with users. For example, the sample map *XXX*MAP that you create in this chapter is used to display existing department records to users for modification or deletion. *XXX*MAP also allows users to add new department records. The *XXX*MAP layout is shown in the diagram below. This sample map allows users to input and display a department's ID number and name, and the ID number of the department head.

XXXMAP layout



```
FUNCTION: ________

                           DEPARTMENT INFORMATION


DEPARTMENT ID .......: ____
           NAME .....: ____________________________________________________
           HEAD ID ..: ____



NEXT RESPONSE:  ________
```

Defining a map: You define the **layout** of a map by defining individual fields on the map. You can define two types of fields:

- **Literal fields** display unchanging literal strings.

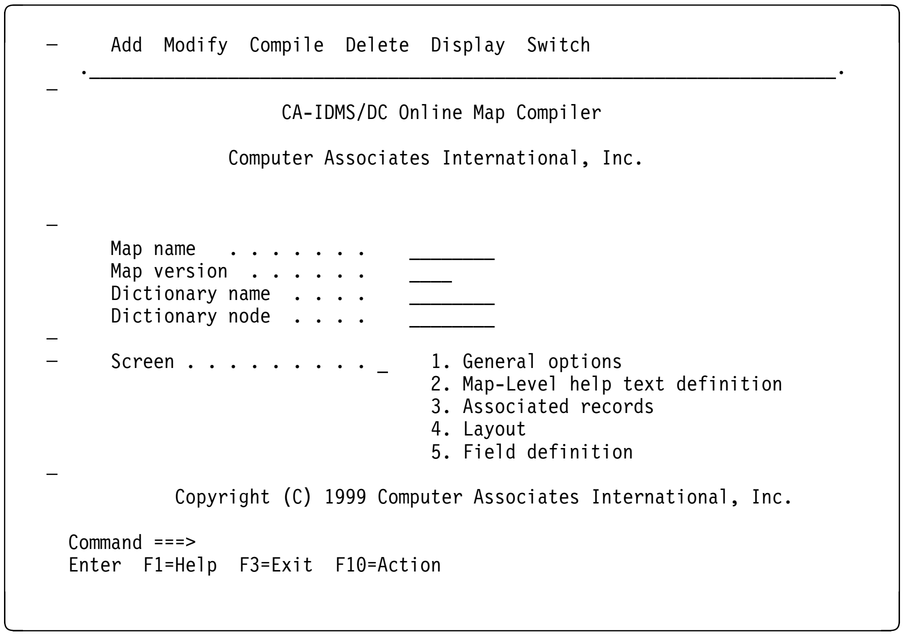
  Titles, instructions, and prompts often are defined as literal fields. For example, the DEPARTMENT INFORMATION title shown in the map layout is a literal field.

- **Variable fields** display stored values and allow users to store new values at runtime.

  For example, the field to the right of the DEPARTMENT ID field on the map layout is a variable field. At runtime, this field displays the ID number for a department record. Additionally, the user can type a new id number into this variable field to modify or add a department record. The last field on the screen, the message field, is also a variable field. At runtime, this field displays a message (message field is blank in this sample screen).

You will use the **online mapping facility (MAPC)** to define map *XXX*MAP shown in the map layout.

A typical MAPC screen is shown below:



```
_     Add  Modify  Compile  Delete  Display  Switch
   .______________________________________________________________________.
_
                    CA-IDMS/DC Online Map Compiler

               Computer Associates International, Inc.


_
      Map name   . . . . . . .    ________
      Map version  . . . . . .    ____
      Dictionary name  . . . .    ________
      Dictionary node  . . . .    ________
_
_     Screen . . . . . . . . . _    1. General options
                                    2. Map-Level help text definition
                                    3. Associated records
                                    4. Layout
                                    5. Field definition
_
            Copyright (C) 1999 Computer Associates International, Inc.

  Command ===>
  Enter  F1=Help  F3=Exit  F10=Action
```

8.3 Instructions

You use MAPC to define maps. To define *XXX*MAP in this chapter, you will:

1. Invoke MAPC
2. Name the map
3. Name the records
4. Create the map with the autopaint facility
5. Modify the layout of the map, if necessary
6. Select fields for further definition
7. Edit the selected fields
8. Compile the map (creating a map load module)

After you compile the map, you can exit from MAPC and optionally display the map you have just defined.

Steps for defining map *XXX*MAP are presented below.

8.3.1 Step 1: Invoke MAPC

You can invoke MAPC from CA-IDMS-DC or CA-IDMS/UCF (DC/UCF) by specifying the task code for MAPC (for example, MAPCT) in response to the prompt presented by DC/UCF. For example, you can invoke MAPC from CA-IDMS/DC as shown:

```
ENTER NEXT TASK CODE:
mapct

                                        Press the ENTER
                                        key to input the -→  [Enter]
                                        task code for
                                        MAPC.
```

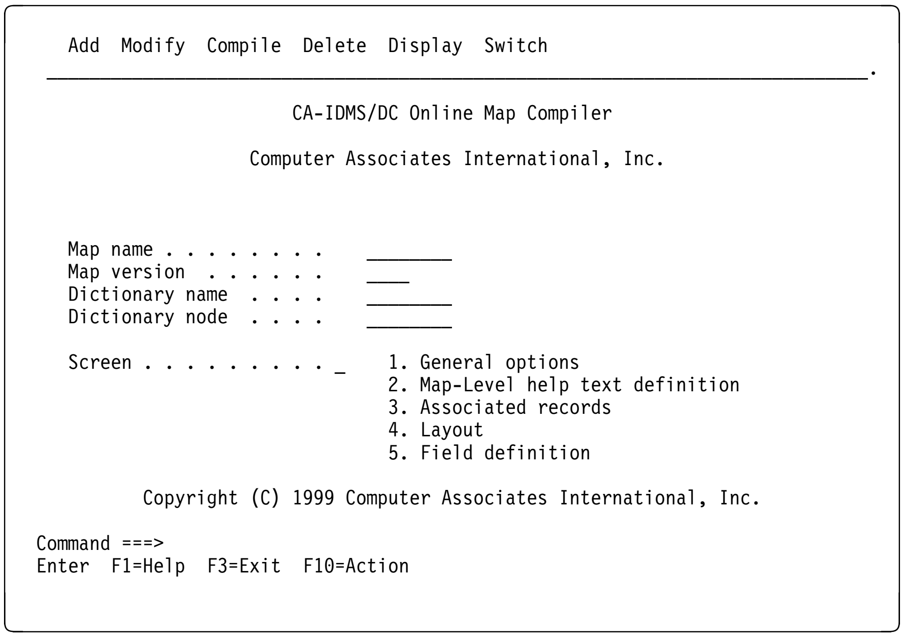
For more information on task codes for CA-ADS development tools, see 6.3, “Application development tools” on page 6-7.

MAPC begins by displaying the Main Menu screen, on which you specify basic information about a map. Use the Main Menu screen to begin defining map *XXX*MAP, as described in Step 2.

8.3.2 Step 2: Name the map

The first screen in an MAPC session is the **Main Menu** screen. A sample Main Menu is shown below:

Sample Main Menu screen



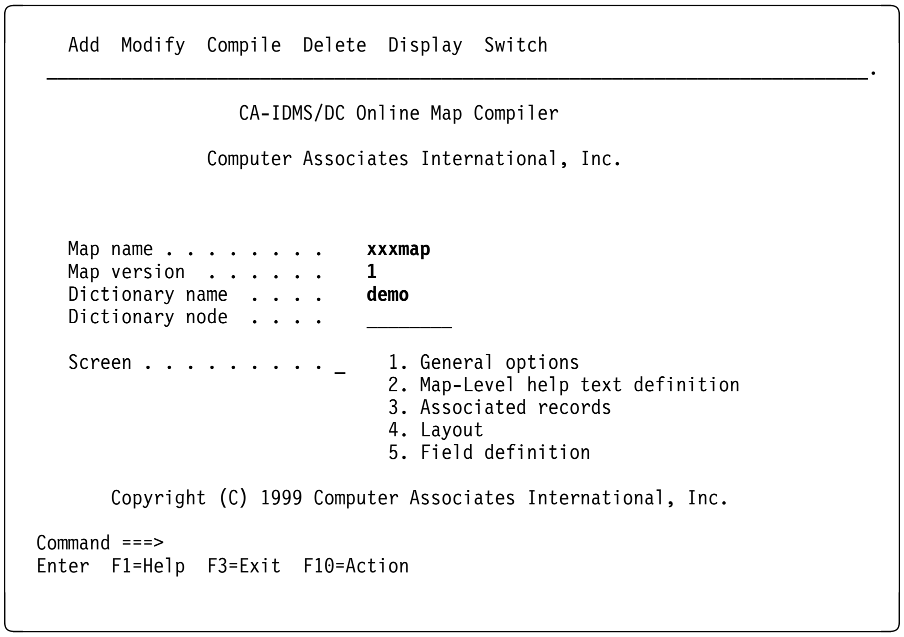
```
    Add  Modify  Compile  Delete  Display  Switch
  _____________________________________________________________________________.

                       CA-IDMS/DC Online Map Compiler

                   Computer Associates International, Inc.



    Map name . . . . . . . .    ________
    Map version  . . . . . .    ____
    Dictionary name  . . . .    ________
    Dictionary node  . . . .    ________

    Screen . . . . . . . . . _     1. General options
                                   2. Map-Level help text definition
                                   3. Associated records
                                   4. Layout
                                   5. Field definition

           Copyright (C) 1999 Computer Associates International, Inc.

 Command ===>
 Enter  F1=Help  F3=Exit  F10=Action
```

Screen prompts: When you begin a map definition, you typically enter information after one or more of the following Main Menu screen prompts:

- **Map name** — You must supply a map name. The name you specify must be unique among all programs. (For example, it cannot be the same name as a dialog.)
- **Map version** — You type the version number of the map after the **Map version** prompt. If not specified, the map version defaults to the version number specified as a dictionary option.
- **Dictionary name** — You must specify the same dictionary (if any) as you specified for your application definition in 7.3.2, "Step 2: Name the application" on page 7-11. The correct dictionary name may already be displayed in this field.
- **Dictionary node** — You must specify the same dictionary node (if any) as you specified for your application definition in Chapter 7. The correct dictionary node may already be displayed in this field.

You specify basic information about map *XXX*MAP on the Main Menu screen.

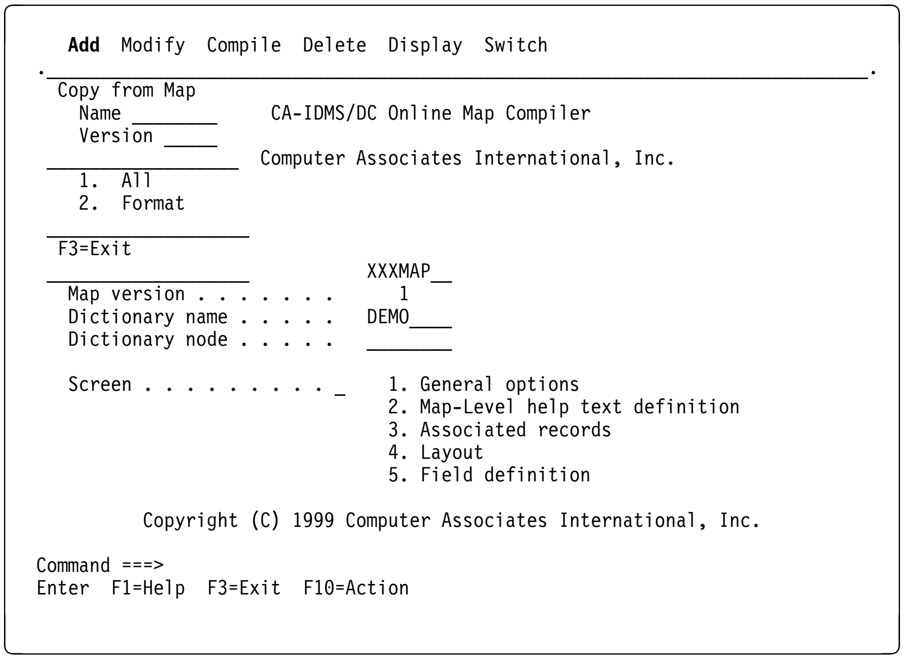
Defining the XXXMAP map: You can use the **tab key** to move the cursor quickly and easily between prompts. Begin defining your application on the Application Definition screen as shown below:



```
   Add  Modify  Compile  Delete  Display  Switch
  _____________________________________________________________________________.

                  CA-IDMS/DC Online Map Compiler

               Computer Associates International, Inc.



   Map name . . . . . . . .    xxxmap
   Map version  . . . . . .    1
   Dictionary name  . . . .    demo
   Dictionary node  . . . .    ________

   Screen . . . . . . . . . _    1. General options
                                 2. Map-Level help text definition
                                 3. Associated records
                                 4. Layout
                                 5. Field definition

       Copyright (C) 1999 Computer Associates International, Inc.

Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

Adding the map: To add the map, position the cursor on the **Add** item on the action bar and press [Enter]. You can position the cursor on **Add** by:

- Tabbing to **Add** and pressing [Enter]
- Pressing [PF10] to move to the action bar and then tabbing to **Add** and pressing [Enter]
- Typing **add** on the command line and pressing [Enter]



```
   Add  Modify  Compile  Delete  Display  Switch
 ._____________________________________________________________________________.
  Copy from Map
    Name ________     CA-IDMS/DC Online Map Compiler
    Version _____
 __________________  Computer Associates International, Inc.
    1.  All
    2.  Format
 ___________________
  F3=Exit
 ___________________           XXXMAP__
   Map version . . . . . . .      1
   Dictionary name . . . . .   DEMO____
   Dictionary node . . . . .   ________

   Screen . . . . . . . . . _    1. General options
                                 2. Map-Level help text definition
                                 3. Associated records
                                 4. Layout
                                 5. Field definition

          Copyright (C) 1999 Computer Associates International, Inc.

Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

Once you have displayed the **Add** action item, press [Enter] to add the map to the dictionary. After you press [Enter], the action is confirmed.

MAPC redisplays the Main Menu screen with a message:

- **Map XXXMAP version 1 has been added** is returned when your definition contains no errors.
- **An error message** is returned if MAPC detects any errors. Read the message to see what problem has occurred. You can type over any errors and then press [Enter] again.

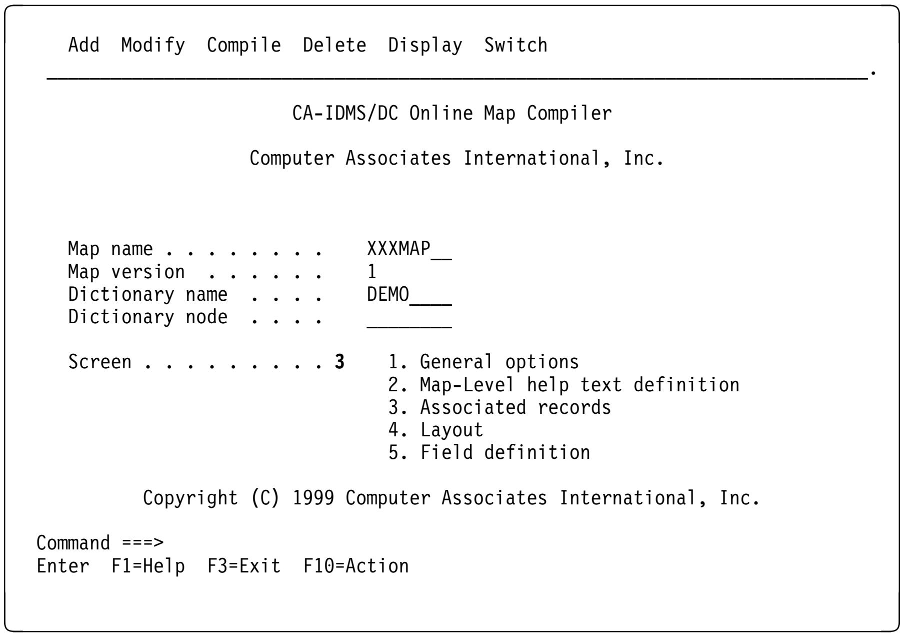
After you provide basic information about the map, you can specify the records you want to use on the map.

8.3.3 Step 3: Name the records

The **Associated Records** screen is used to enter the schema or work records to be used by the map, and optionally specifies role names for records.

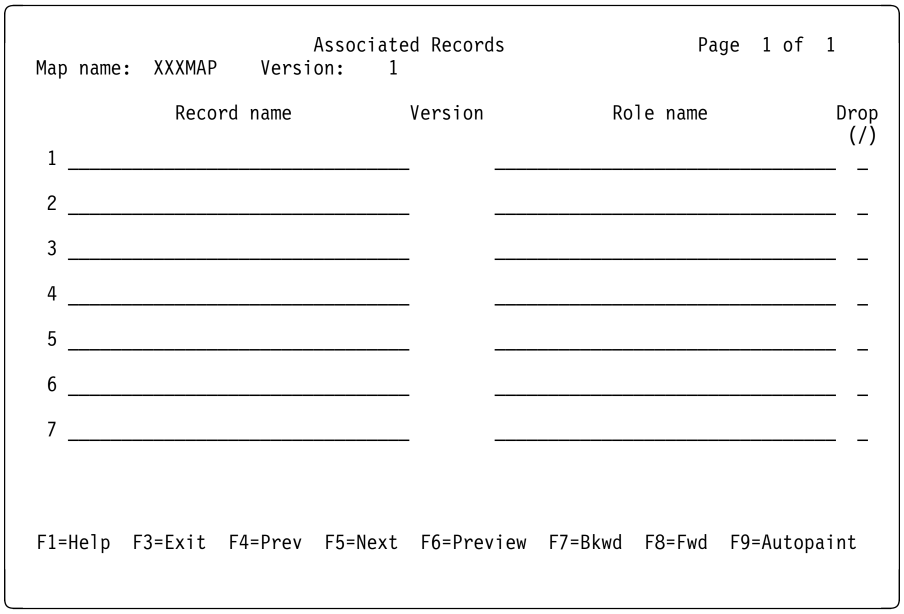
The autopaint feature is invoked from this screen. The autopaint feature automatically paints the map based on the elements you select.

You access the Associated Records screen from the Main Menu by entering the number **3** next to the **Screen** prompt and pressing [Enter].



```
   Add  Modify  Compile  Delete  Display  Switch
 _____________________________________________________________________________.

                        CA-IDMS/DC Online Map Compiler

                    Computer Associates International, Inc.



   Map name . . . . . . . .    XXXMAP__
   Map version  . . . . . .    1
   Dictionary name  . . . .    DEMO____
   Dictionary node  . . . .    ________

   Screen . . . . . . . . . 3    1. General options
                                 2. Map-Level help text definition
                                 3. Associated records
                                 4. Layout
                                 5. Field definition

          Copyright (C) 1999 Computer Associates International, Inc.

Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

Sample Associated Records screen



```
                          Associated Records                Page  1 of  1
Map name:  XXXMAP    Version:    1

             Record name           Version           Role name           Drop
                                                                          (/)
 1 ________________________________        ________________________________  _

 2 ________________________________        ________________________________  _

 3 ________________________________        ________________________________  _

 4 ________________________________        ________________________________  _

 5 ________________________________        ________________________________  _

 6 ________________________________        ________________________________  _

 7 ________________________________        ________________________________  _



F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F7=Bkwd  F8=Fwd  F9=Autopaint
```

Screen prompts: On the Associated Records screen, you name the records that will be used on the map:

- **Record name** — You associate existing database and work records with the map by naming the records in the lines below the **Record name** prompt.

  Associating a record with a map allows the map to display and store data for that record. Records associated with maps sometimes are referred to as **map records** to indicate that the records define data used by the map.

  Records contain record **elements**, which define data. In relational terminology, a record is a data table that contains columns defining data to be stored. An example of a record is the DEPARTMENT database record. One of the elements in the DEPARTMENT record is DEPT-NAME-0410. This element stores department names. The layout of the sample DEPARTMENT record is provided in Appendix C, "Layout of the DEPARTMENT Record" on page C-1.

- **Version** — You type the version number of the record below the **Version** prompt.

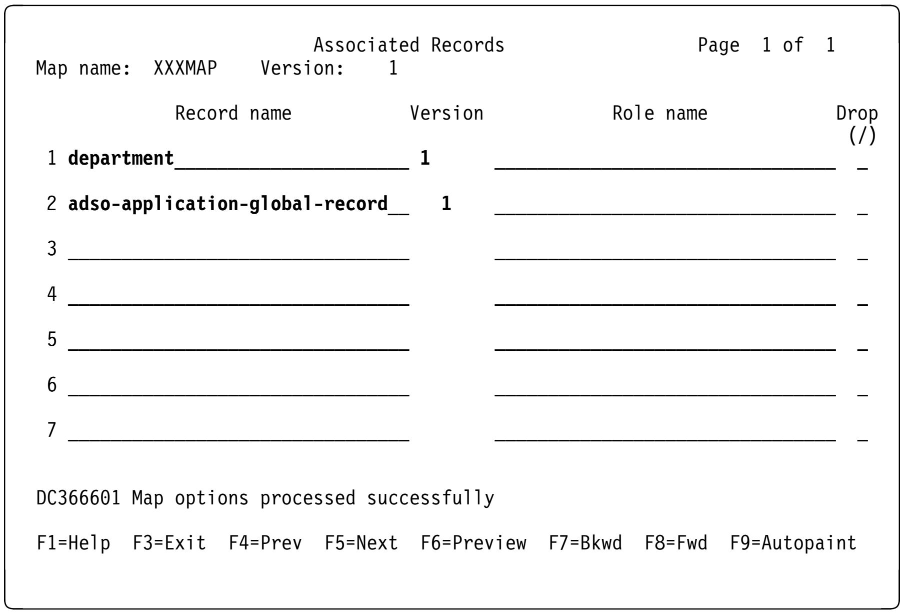
  For example, the demonstration database at your site might have a few different versions of the DEPARTMENT record, each reserved for specific testing or development purposes. In this case, each different version of the record has a unique version number (for example: 1, 2, or 100). To ensure that MAPC uses the correct version of the record, you specify the record version number along with the record name.

Records associated with XXXMAP: The following table lists the records you will associate with map *XXX*MAP.

Record	Purpose
DEPARTMENT Version: 1 *	A database record in the demonstration database. DEPARTMENT includes elements for department ID number, department name, and the employee ID of the department head.
ADSO-APPLICATION-GLOBAL-RECORD Version: 1	A special CA-ADS record that contains information about the application at runtime. For example, this record includes a record element that at runtime contains the name of the currently executing function.

* A different version of the DEPARTMENT record may be provided for use at your site.

Associating records with XXXMAP



```
                        Associated Records                Page  1 of  1
Map name:  XXXMAP    Version:    1

             Record name          Version            Role name            Drop
                                                                           (/)
  1 department______________________ 1       ________________________________ _

  2 adso-application-global-record__   1     ________________________________ _

  3 ________________________________         ________________________________ _

  4 ________________________________         ________________________________ _

  5 ________________________________         ________________________________ _

  6 ________________________________         ________________________________ _

  7 ________________________________         ________________________________ _


DC366601 Map options processed successfully

F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F7=Bkwd  F8=Fwd  F9=Autopaint
```

After you have entered the names of the records on the Associated Records screen, press [PF9] to use the autopaint facility to create the map.

8.3.4 Step 4: Create the map with the autopaint facility

You can create a map either manually or through the autopaint facility.

To create a map manually, you would name the records on the Associated Records screen and then place each literal and variable field explicitly on the map using the Layout screen. Then you would further define each variable field to associate a record element with that field.

►► For further information, see *CA-IDMS Mapping Facility*.

The autopaint facility quickly creates a standard map layout based on the records you have named on the Associated Records screen. The autopaint facility is useful for maps which require little or no explicit screen placement.

You will use the autopaint facility to create the *XXX*MAP for the Department application.

The first step in using the autopaint facility to create a map is to go to the **Automatic Screen Painter** screen to identify the fields you want to have displayed on the map. Here you determine what record elements will be displayed on the screen.

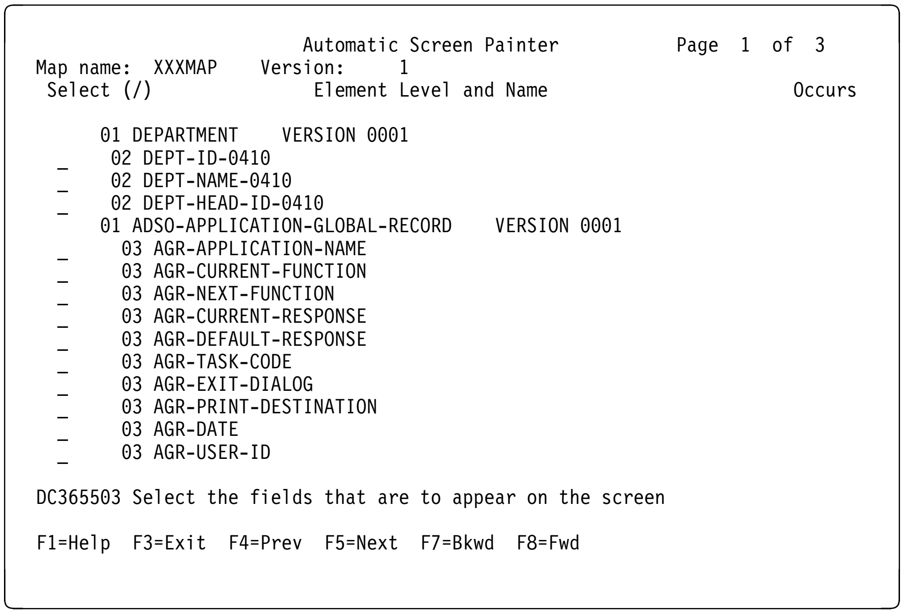
A **record element** is a data definition that is contained in a record.

For example, the DEPARTMENT record contains a record element for a department's name, DEPT-NAME-0410.

If the dialog retrieves a department record from the database, the record is temporarily stored in variable storage. The record name in variable storage is automatically displayed when the map is displayed. If the user enters a valid department name in this map field, the data is automatically moved into variable storage. It can then be saved in the database or used to access other data, depending on the dialog code.

To access the **Automatic Screen Painter** screen, press [PF9] from the Associated Records screen.

Sample Automatic Screen Painter screen



```
                         Automatic Screen Painter          Page  1  of  3
Map name:  XXXMAP    Version:     1
 Select (/)               Element Level and Name                       Occurs

      01 DEPARTMENT    VERSION 0001
  _    02 DEPT-ID-0410
  _    02 DEPT-NAME-0410
  _    02 DEPT-HEAD-ID-0410
      01 ADSO-APPLICATION-GLOBAL-RECORD    VERSION 0001
  _     03 AGR-APPLICATION-NAME
  _     03 AGR-CURRENT-FUNCTION
  _     03 AGR-NEXT-FUNCTION
  _     03 AGR-CURRENT-RESPONSE
  _     03 AGR-DEFAULT-RESPONSE
  _     03 AGR-TASK-CODE
  _     03 AGR-EXIT-DIALOG
  _     03 AGR-PRINT-DESTINATION
  _     03 AGR-DATE
  _     03 AGR-USER-ID

DC365503 Select the fields that are to appear on the screen

F1=Help  F3=Exit  F4=Prev  F5=Next  F7=Bkwd  F8=Fwd
```

Screen prompts: On the Automatic Screen Painter screen, you can select the fields that will be used on the map. The fields will be listed under the appropriate record name. Use a nonblank character to select the fields you want displayed.

- **Element level and name** — Each element within a record listed on the Associated Records screen will be displayed. Each element is associated with its level number as defined in the dictionary.
- **Occur** — If the element is a repeating element, you can define the occurrence number in this column.

There can be multiple pages of elements. Press [PF8] to continue to the next page.

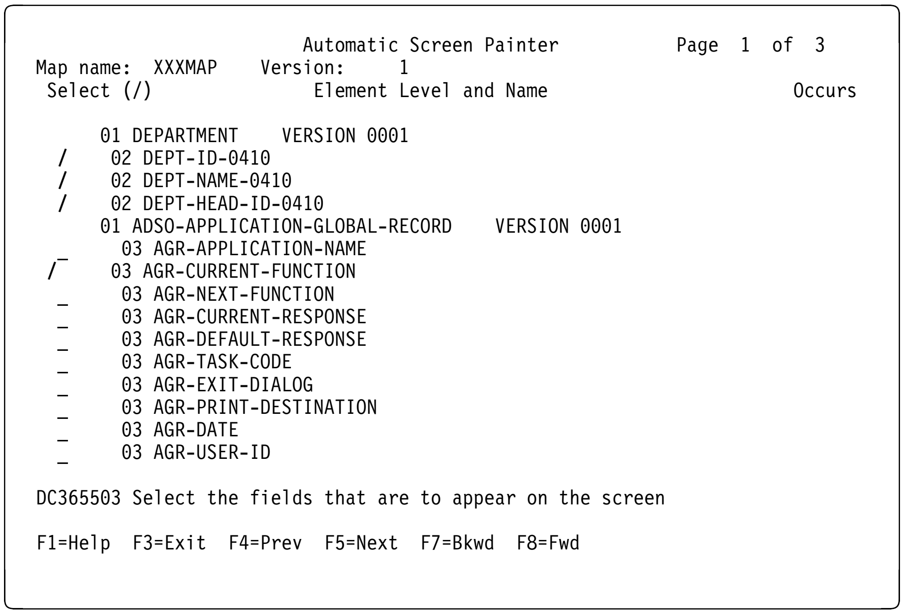
Fields on XXXMAP: The following table lists the records and fields to be displayed on *XXX*MAP.

Record	Fields
DEPARTMENT	DEPT-ID-0410 DEPT-NAME-0410 DEPT-HEAD-ID-0410
ADSO-APPLICATION-GLOBAL-RECORD	AGR-CURRENT-FUNCTION

Note:

The database does not have to be defined before you create the prototype map layout. If the database were *not* already defined, you would define all literal fields manually. (See Appendix C, "Layout of the DEPARTMENT Record" on page C-1)

Selecting fields for use with XXXMAP



```
                         Automatic Screen Painter          Page  1  of  3
Map name:  XXXMAP    Version:     1
 Select (/)              Element Level and Name                       Occurs

      01 DEPARTMENT    VERSION 0001
   /    02 DEPT-ID-0410
   /    02 DEPT-NAME-0410
   /    02 DEPT-HEAD-ID-0410
      01 ADSO-APPLICATION-GLOBAL-RECORD    VERSION 0001
   _     03 AGR-APPLICATION-NAME
  /¯    03 AGR-CURRENT-FUNCTION
   _     03 AGR-NEXT-FUNCTION
   _     03 AGR-CURRENT-RESPONSE
   _     03 AGR-DEFAULT-RESPONSE
   _     03 AGR-TASK-CODE
   _     03 AGR-EXIT-DIALOG
   _     03 AGR-PRINT-DESTINATION
   _     03 AGR-DATE
   _     03 AGR-USER-ID

DC365503 Select the fields that are to appear on the screen

F1=Help  F3=Exit  F4=Prev  F5=Next  F7=Bkwd  F8=Fwd
```

After you have selected the names of the fields on the Automatic Screen Painter screen, press [PF5] to paint the map.

Note: You can press [Enter] first if you want the screen to be redisplayed to check your choices.

If there are multiple pages of elements, you would use [PF8] to move to subsequent screens for further selection.

The autopainted screen will be displayed on the **Layout** screen. Now you can modify the placement of the fields and request that some fields be further defined.

8.3.5 Step 5: Modify the map layout

You use the **Layout** screen to modify the layout of fields on a map. There are two types of fields on the screen:

- Map **literal fields** display predefined literal strings at runtime.
- Map **variable fields** display stored values and allow users to store new values at runtime.

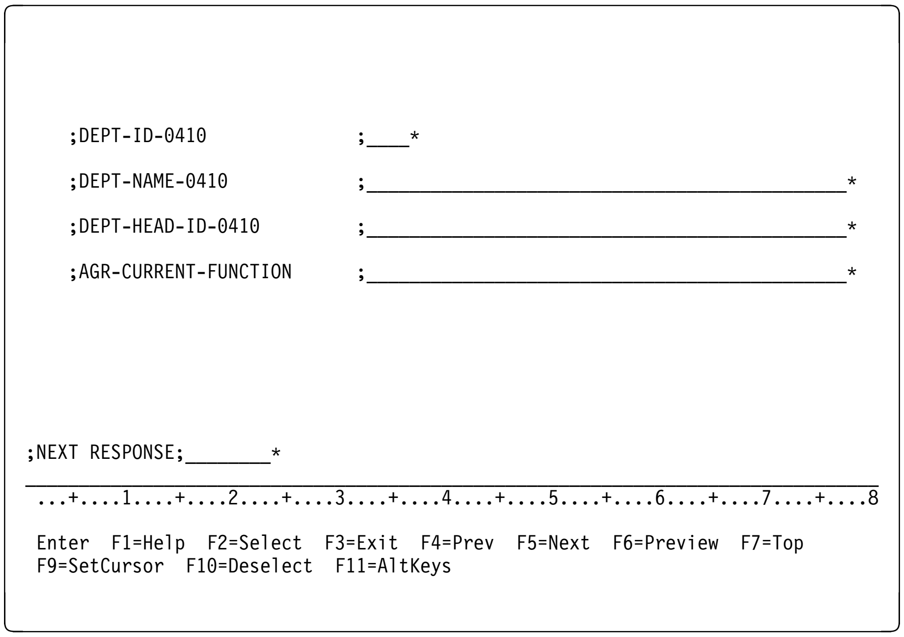
When you modify map *XXX*MAP in this chapter, you will:

- Add new literal fields
- Modify literal fields
- Change the placement of variable and literal fields

Accessing the Layout screen: Once you have pressed [PF5] from the Automatic Screen Painter screen, the **Layout** screen is displayed to you with the automatically-created map presented. Fields at the bottom of the screen show key functions and a scale.

To reveal the hidden portion of the screen, press [PF8].

Sample Layout screen with additional fields displayed



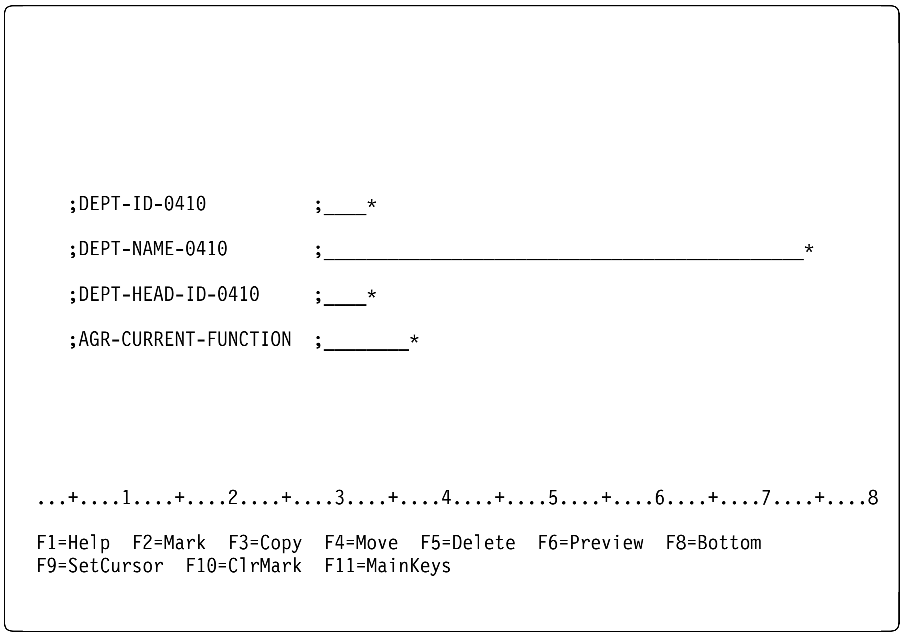
```
    ;DEPT-ID-0410             ;____*

    ;DEPT-NAME-0410           ;_____________________________________________*

    ;DEPT-HEAD-ID-0410        ;_____________________________________________*

    ;AGR-CURRENT-FUNCTION     ;_____________________________________________*





;NEXT RESPONSE;________*
________________________________________________________________________________
 ...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8

 Enter  F1=Help  F2=Select  F3=Exit  F4=Prev  F5=Next  F6=Preview  F7=Top
 F9=SetCursor  F10=Deselect  F11=AltKeys
```

Note: Although you did not specify a response or message field, ADSA provided both. Since you associated the global record with this map, ADSA provided AGR-MAP-RESPONSE and AGR-MESSAGE as the response and message variable fields. If you had not associated the ADSO-APPLICATION-GLOBAL-RECORD with this map, ADSA would have provided $RESPONSE and $MESSAGE as the response and message variable fields.

Notice that the field mark for the message field is in column 80 so that the message will begin in column 1 of the following line.

Press [PF11] to reveal the alternate PF key set used for tailoring the screen.

Sample Layout screen with alternate set of PF keys displayed



```
   ;DEPT-ID-0410        ;____*

   ;DEPT-NAME-0410      ;_____________________________________________*

   ;DEPT-HEAD-ID-0410   ;____*

   ;AGR-CURRENT-FUNCTION ;________*




...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8

F1=Help  F2=Mark  F3=Copy  F4=Move  F5=Delete  F6=Preview  F8=Bottom
F9=SetCursor  F10=ClrMark  F11=MainKeys
```

When you first display the Layout screen for an existing map, each field on the map is preceded by a start-field character, as shown above. While using the Layout screen, you can select a field for editing by:

- Pressing [PF2] while on the field you want to select (with the main set of function keys displayed at the bottom of the screen)

  or

- Typing a select-field character (%) in place of the start-field character for the field.

You use the start-field and select-field characters based on the following guidelines:

- The **start-field character** (default is ; or &lbr.) defines the start of a field on the Layout screen.
- The **select-field character** (default is %) defines the start of a field and simultaneously selects the field for editing.

You use the Layout screen to modify the layout of map *XXX*MAP and to select fields on the map for further editing:

Selecting multiple fields to edit: You can mark two fields with [PF2] to select either all literal fields or all data fields in the area bounded by the two fields you mark.

If the first field you marked with [PF2] was a literal field, all literal fields between the two marked fields are selected. If the first field you marked with [PF2] was a data field, all data fields between the two marked fields are selected.

Adding literal fields: You place each field on the Layout screen in the following manner:

1. **Position the cursor** by using any of the cursor movement keys.
2. **Type a start-field character** for each literal and variable field.

Note: In this document, the start-field character is shown as a ";". Do not confuse this with a semi-colon.

On the Layout screen, the start-field character signals the start of a field. For example:

```
Sample start-field
character.
|
↓
;sample literal field
 ↑
 |
 The field itself starts in the column that
 immediately follows the start-field character.
```

At runtime, the start-field character is not shown to users with 3270-type terminals. Instead, each field is preceded by a nondisplayable **attribute byte**. The attribute byte specifies the runtime characteristics of the field, such as input restrictions and display intensity.

The default start-field character for the Layout screen is:

- For **IBM-type terminals**, the field mark character (;)

  Note: The field mark is *not* the same as the semicolon character. To type a field mark, you press the FIELD MARK key.

- For **Siemens-type terminals,** the left brace character (&lbr.)

  Note: The start-field character for the Layout screen is defined at system-generation time, and can vary from site to site.

- **Type the literal string** (for literal fields only) after the start-field character.

  Note: To add a variable field, you need only type the start-field character. (See Chapter 11, "Modifying a Map Using MAPC" on page 11-1.)

Changing the content of a literal field: To change the content of an existing literal field, type characters or spaces over the fields that you want to change.

Note: Use the ERASE EOF key only if you want to erase *everything* that can be seen on the Layout screen starting at the current cursor position.

Moving fields, lines, and blocks: You can move fields or groups of fields:

- **Moving one field** — Move the cursor to the field you want to move and press [PF2] with the alternate set of function keys displayed at the bottom of the screen. This marks the field.
- **Moving a line** — Move the cursor to the starting position of the block you want to move and press [PF2] **twice**. This marks the line.

- **Moving a block** — Move the cursor to the starting position of the block you want to move and press [PF2]. Move the cursor to the ending point of the block and press [PF2] again. This marks a block.

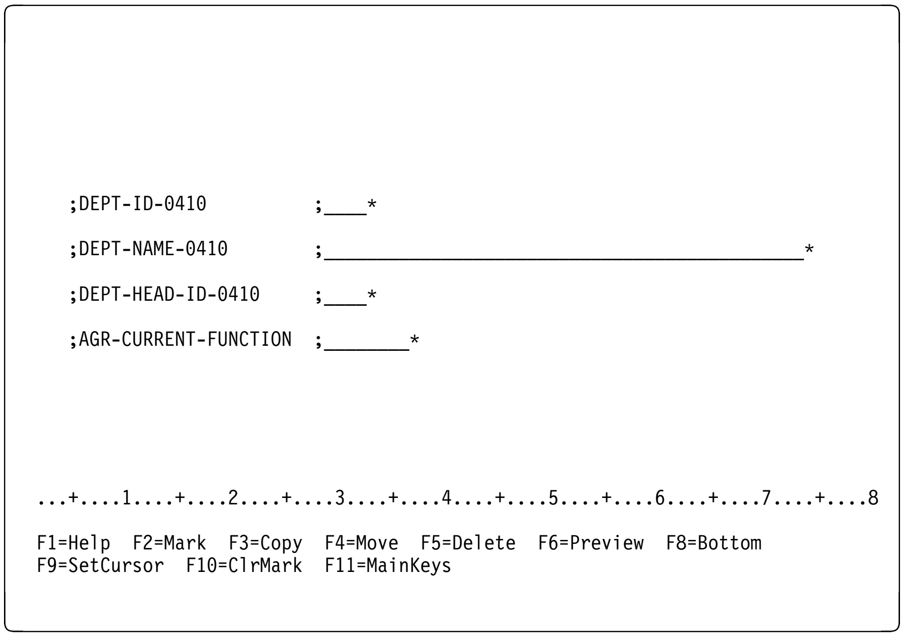
Move the cursor to the desired target location for the field, line, or block, and press [PF4].

Deleting fields, lines, and blocks: Mark the field, line, or block. Then press [PF5].

Copying fields, lines, and blocks: Mark the field, line, or block. Then press [PF3].

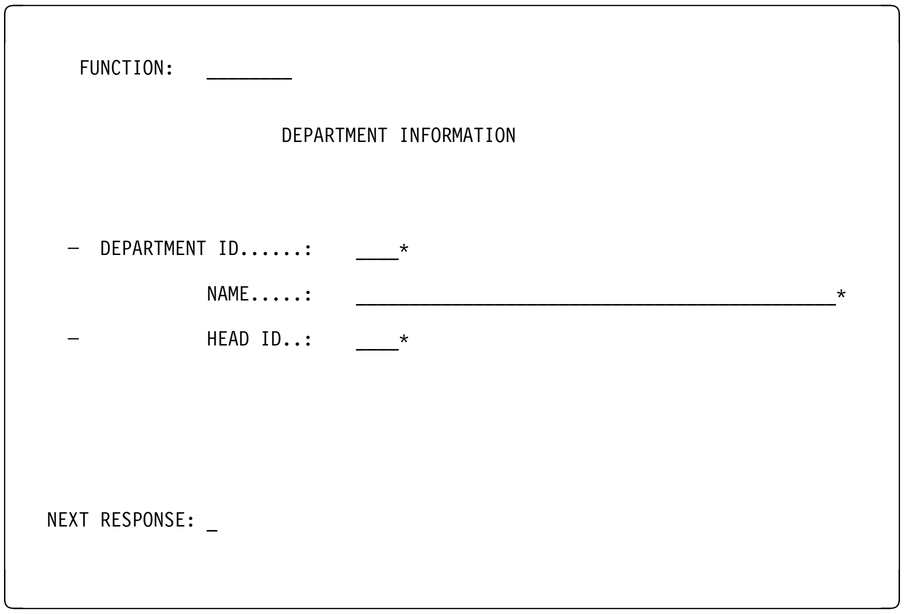
When you copy literal or variable fields, the complete definition of the literal or variable field (including attributes, etc.) is copied. Copying a data field that occurs increments the subscript to the next available value.

Modifying the map layout for XXXMAP: For map *XXX*MAP, you selected four variable fields that are now shown on the Layout screen. These are associated with four literal fields. Each literal and variable field is shown with the **field mark** (;) used as the start-field character.



```
   ;DEPT-ID-0410         ;____*

   ;DEPT-NAME-0410       ;_____________________________________________*

   ;DEPT-HEAD-ID-0410    ;____*

   ;AGR-CURRENT-FUNCTION ;________*




...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8

F1=Help  F2=Mark  F3=Copy  F4=Move  F5=Delete  F6=Preview  F8=Bottom
F9=SetCursor  F10=ClrMark  F11=MainKeys
```

You are going to modify this map so that it looks like the following layout:



```
   FUNCTION:   _______

                     DEPARTMENT INFORMATION


  -  DEPARTMENT ID......:    ____*
             NAME.....:    _____________________________________________*
  -          HEAD ID..:    ____*




 NEXT RESPONSE: _
```

Your modifications involve:

- Changing the AGR-CURRENT-FUNCTION literal field to FUNCTION
- Moving the FUNCTION literal and variable fields
- Adding a title
- Modifying the remaining literal fields

Modifying the FUNCTION fields for XXXMAP: The AGR-CURRENT-FUNCTION fields (literal and variable) should be placed in the upper left corner of the screen and the literal changed to FUNCTION according to the sample screen.

To change the FUNCTION field:

1. Overtype the literal field (leaving the field mark) with the word FUNCTION
2. Mark the **variable** field
3. Move the variable field closer to the FUNCTION literal field
4. Mark the line containing the literal and variable fields
5. Move the line to the upper left corner

Add the title

1. Place the cursor where you want the title to begin.
2. Type a field mark
3. Type the title, DEPARTMENT INFORMATION

Modify the remaining literal fields: There are three literal fields relating to the DEPARTMENT record. Modify these fields so that they match the *XXX*MAP screen shown above.

1. Overtype the literal fields with the appropriate words
2. Mark each field that needs to be moved
3. Move the field

Remember that there is more room for screen layout hidden at the bottom of the screen. To see this hidden area, press [PF8].

The completed screen should look like the one below.

```
;FUNCTION:;

                        ;DEPARTMENT INFORMATION

;DEPARTMENT ID .......:;
           ;NAME .....:;
           ;HEAD ID ..:;







;NEXT RESPONSE: ;
                                                                               ;
```

After you press [Enter], MAPC redisplays the Layout screen so that you can inspect the screen for errors. At this point, it is a good idea to verify that:

- You have preceded each **literal field** with a start-field character.
- You have defined each **variable field** with a start-field character, including the field that starts on the bottom right-hand margin of the screen and wraps around to the last line on the screen.

Correcting errors: If you find any mistakes in the map layout, you can correct the Layout screen in either of the following ways:

- **To change a few fields**, type over the characters that you want to change and press [Enter] again.
- **To erase all fields that you just placed on the screen**, press the CLEAR key. If you press CLEAR, you must then place fields on the Layout screen again, as described earlier.

When you are satisfied with the Layout screen, press [PF11] to return to the main keys. You can now go on to Step 6, where you will select fields for further definition.

8.3.6 Step 6: Select fields for further definition

In Step 5, you modified the position of fields on your map and redefined some of the literal fields.

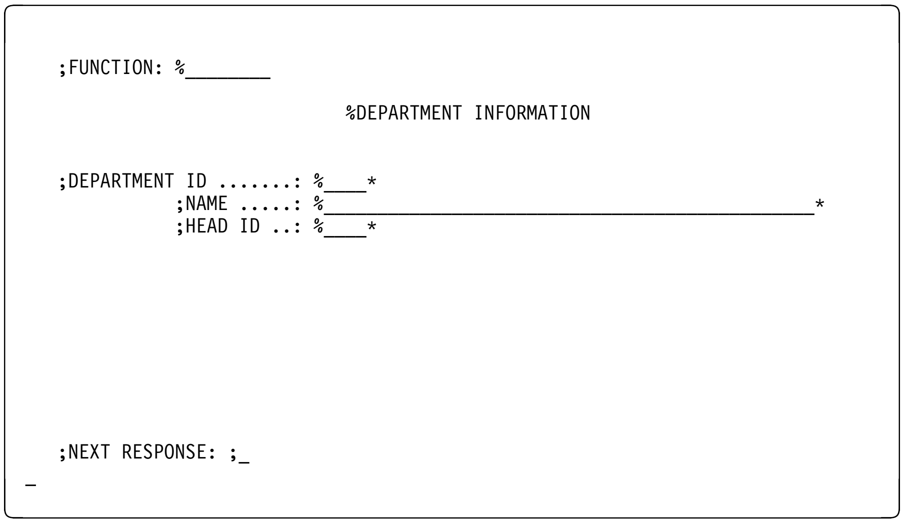
At this point, literal fields (for example, DEPARTMENT ID) are fully defined, although you can modify their definitions at any time.

Variable fields may not be fully defined. In this step, you will select fields for further definition. You will edit the field definitions in Steps 7 and 8.

Selecting fields: While on the Layout screen, you can select fields for further definition. To do this, you press [PF2] once while the cursor is on the field you want to select. Pressing [PF2] marks a field for selection. (Alternatively, you can overtype the start-field character with a percent sign - %.)

Select XXXMAP fields: The *XXX*MAP fields you need to further define are:

- The FUNCTION **variable** field
- The DEPARTMENT INFORMATION **literal** field
- The DEPARTMENT ID **variable** field
- The NAME **variable** field
- The HEAD ID **variable** field
- The message **variable** field



```
  ;FUNCTION: %________

                              %DEPARTMENT INFORMATION


  ;DEPARTMENT ID .......: %____*
             ;NAME .....: %______________________________________________*
             ;HEAD ID ..: %____*








  ;NEXT RESPONSE: ;_
_
```

Select the fields for further definition. Then press [PF5] to continue to the Literal Definition and Field Definition screens shown below. MAPC will bring up the appropriate Literal Definition or Field Definition screen depending on the fields you selected on the Layout screen. Pressing [PF5] will bring you to the next definition screen in order of your selection.

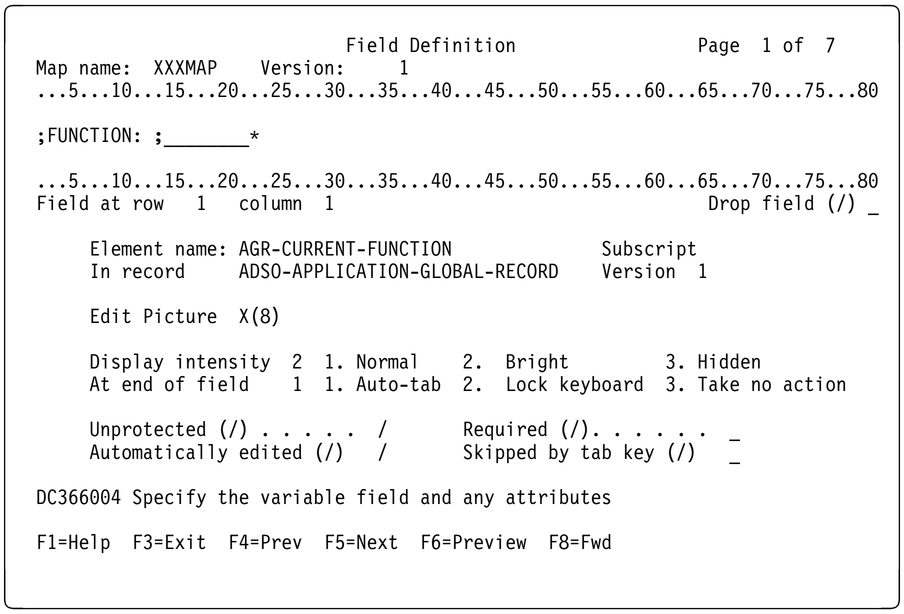
This chapter separates the discussion of variable and literal fields, but remember that MAPC will intermix the two.

8.3.7 Step 7: Edit variable fields

In this step, you will edit a variable field's definition to determine what characteristics the field will have at runtime. You access the Field Definition screen from the Layout screen by pressing [PF5] after you have selected fields from the screen, or by pressing [PF5] from another definition screen (either Field or Literal Definition).

There are seven pages of data field screens. Navigate through these pages using [PF7] or [PF8], or move directly to the desired page by overtyping the page number.

Sample Field Definition screen - page 1



```
                             Field Definition                Page  1 of  7
Map name:  XXXMAP    Version:     1
...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80

;FUNCTION: ;_______*

...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80
Field at row   1   column  1                                 Drop field (/) _

     Element name: AGR-CURRENT-FUNCTION              Subscript
     In record     ADSO-APPLICATION-GLOBAL-RECORD    Version  1

     Edit Picture  X(8)

     Display intensity  2  1. Normal    2.  Bright        3. Hidden
     At end of field    1  1. Auto-tab  2.  Lock keyboard  3. Take no action

     Unprotected (/) . . . . .  /      Required (/). . . . . .  _
     Automatically edited (/)   /      Skipped by tab key (/)   _

DC366004 Specify the variable field and any attributes

F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F8=Fwd
```

Notice that the Field Definition screen shows the field to be defined plus two scales to help you position the field.

You use the Field Definition screen to edit one map field at a time. After you edit a field (using [PF8] if you need to go to further pages), press [PF5]. MAPC displays either the Field Definition screen or the Literal Definition screen for the next field that you've selected. When you've edited all selected fields, MAPC returns you to the Main Menu screen.

Note: Pressing [Enter] on any screen causes it to redisplay so that you can make changes if you want.

Prompts: In this step, you use the following prompts on page 1 of the Field Definition screen:

- **Element name** — Name of a record element or special system field to be displayed and input in the variable map field.

  When you autopainted the map, you selected the record elements you wanted displayed in the variable fields on that map (such as the department-name record element for the variable map field that follows the literal FUNCTION field). These element names are displayed on the Field Definition screen. At runtime, that variable map field displays and allows users to input department names.

  If you were to create a map manually, you would have to specify the element names on this screen at this time.

  ►► For further information, see *CA-IDMS Mapping Facility*.

  A **special system field** is a field that has a reserved use in an CA-ADS application.

  For example, $MESSAGE is a special system field that contains messages (such as error messages) at runtime. A variable field associated with $MESSAGE can display those messages to the user.

- **Edit Picture** — If you autopaint this map, the length of the field (as defined in the dictionary) will be displayed here. If you create this map manually, the length of the field (as defined in the dictionary) will be displayed here **after you press** [Enter].

  You can create an external picture used for display by entering a different value in **Edit picture**. For example, you might want to change 9(4).99 to $9(4).99 for a monetary value for display purposes.

- **Display intensity** — You can specify the runtime display intensity for the variable map field.

  The default, **1 (Normal)**, causes the field to be displayed at normal intensity. You specify **2 (Bright)** to make a field display at bright intensity or **3 (Hidden)** to make a field invisible to the user.

- **At end of field** — You can specify whether the user is restricted from typing beyond the end of the variable map field.

  The default, **1 (Auto-tab)**, specifies that the field is explicitly delimited. In this case, the user cannot type beyond the end of the field. The cursor will skip to the next unprotected field when the user fills the current field with characters.

  2 (Lock keyboard) causes the keyboard to lock when the user attempts to enter data beyond the end of the field.

  3 (Take no action) specifies that the field is not explicitly delimited. In this case, the user can type beyond the end of the field (although excess characters are truncated on input).

- **Unprotected (/)** — You can specify whether the user can enter data into the variable map field. Spacing over the slash (/) indicates that the map field is protected and restricts the user from changing the contents of the field.
- **Required (/)** — You can specify whether the user must enter data into the variable map field. Entering a nonblank character indicates that data must be entered into the field before the map data will be processed.
- **Automatically edited (/)** — You can enable the automatic data editing feature of CA-ADS.

  For map *XXX*MAP, you will enable this feature for numeric fields (such as the field that displays department numbers) to make them readable.
- **Skipped by tab key (/)** — You can specify that that tab key will not stop on this map field.

▸▸ For more information on using Field Definition screen prompts, see the *CA-IDMS Mapping Facility*.

XXXMAP field specifications: The following table summarizes the specifications that you will make when you edit map fields in this step. You will edit each field definition as indicated in this table.

Location of field on map	Purpose of field	Specifications for field
After FUNCTION literal field (in the upper left corner)	Displays the name of the application function being executed at runtime	Protected Bright display
DEPARTMENT INFORMATION literal field	Displays the title for the screen	Bright display
After DEPARTMENT ID literal field	Displays a department's unique ID number	Auto-tab Automatically edited
After NAME literal field	Displays a department's name	Auto-tab Pad character - space[1]
After HEAD ID literal field	Displays the ID number for the head of the department	Auto-tab Automatically edited
Last field on the map (on the bottom right side)	Displays runtime messages to the use	Element name: AGR-MESSAGE Length: 80 bytes [1] The pad character is defined on page 2 of the Field Definition screen.

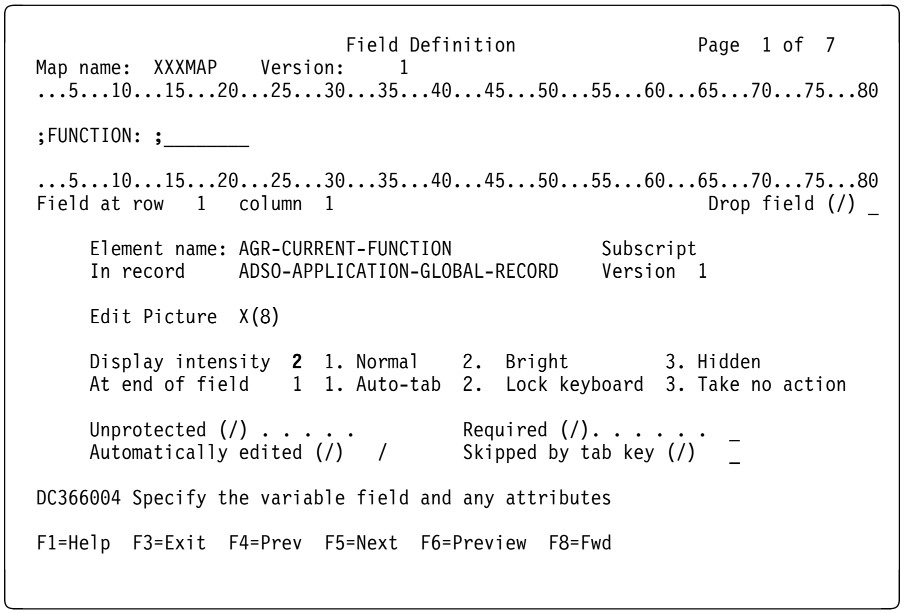
Modifying XXXMAP: Modify the variable field shown on the Field Definition screen (AGR-CURRENT-FUNCTION).

Note: The FUNCTION literal field will not be available for modification because you did not select it on the Layout screen.

Notice that:

- The record element (AGR-CURRENT-FUNCTION) is already displayed for the element name.
- The position of the variable field is shown.
- The edit picture shows the actual length of the function variable field.
- Several defaults are indicated.

To enter the field specifications shown in the preceding table, enter **2** following the **Display intensity** prompt to indicate bright. Space over the slash following **Unprotected** to make this a protected field.



```
                             Field Definition                Page  1 of  7
Map name:  XXXMAP    Version:     1
...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80

;FUNCTION: ;________

...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80
Field at row   1   column  1                                   Drop field (/) _

     Element name: AGR-CURRENT-FUNCTION              Subscript
     In record     ADSO-APPLICATION-GLOBAL-RECORD    Version  1

     Edit Picture  X(8)

     Display intensity  2  1. Normal    2.  Bright         3. Hidden
     At end of field    1  1. Auto-tab  2.  Lock keyboard  3. Take no action

     Unprotected (/) . . . . .         Required (/). . . . . .  _
     Automatically edited (/)   /      Skipped by tab key (/)   _

DC366004 Specify the variable field and any attributes

F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F8=Fwd
```

After you press [Enter], MAPC redisplays the Field Definition screen with a message:

- **Map options processed successfully** is returned when the definition contains no errors.
- **An error message** is returned when MAPC detects an error in your definition. In this case, read the message to determine the problem. You can type over any errors and press [Enter] again.

When the current field definition is correct, press [PF5] to see the next definition to be enhanced.

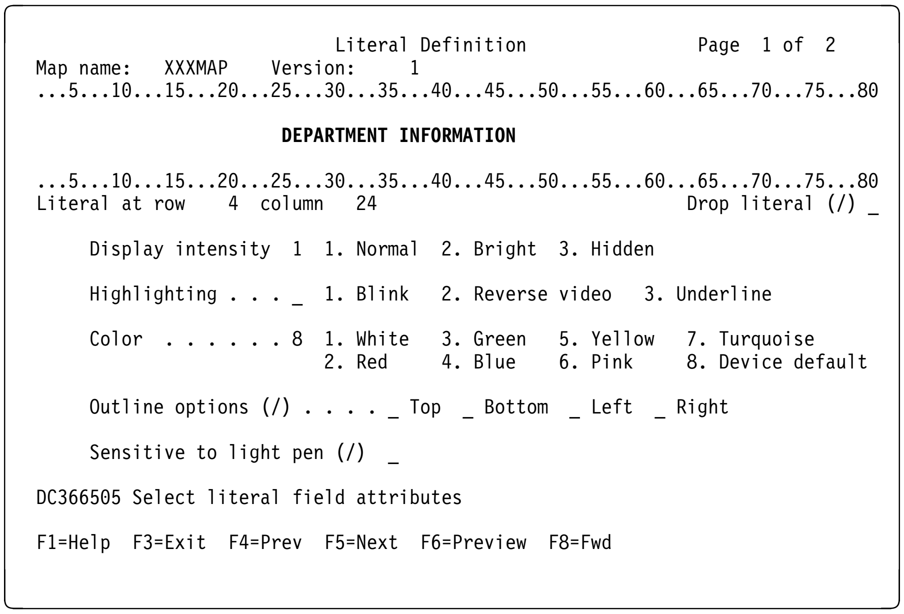
8.3.8 Step 8: Edit literal fields

The next field selected on the Layout screen was the title, DEPARTMENT INFORMATION. When you press [PF5] from the previous Field Definition screen, the **Literal Definition** screen is displayed.

In this step, you will edit a literal field's definition to determine what the field will look like at runtime. You access the Literal Definition screen from the Layout screen by pressing [PF5] after you have selected fields from the screen, or by pressing [PF5] from another definition screen (either Field or Literal Definition).

There are two pages of data field screens. Navigate between these screens using [PF7] or [PF8], or move directly to the desired page by overtyping the page number.

Sample Literal Definition screen - page 1



```
                            Literal Definition                Page  1 of  2
Map name:   XXXMAP    Version:     1
...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80

                       DEPARTMENT INFORMATION

...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80
Literal at row    4  column   24                             Drop literal (/) _

     Display intensity  1  1. Normal  2. Bright  3. Hidden

     Highlighting . . . _  1. Blink   2. Reverse video   3. Underline

     Color  . . . . . . 8  1. White   3. Green   5. Yellow   7. Turquoise
                           2. Red     4. Blue    6. Pink     8. Device default

     Outline options (/) . . . . _ Top  _ Bottom  _ Left  _ Right

     Sensitive to light pen (/)  _

DC366505 Select literal field attributes

F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F8=Fwd
```

Notice that each definition screen shows the field to be defined plus two scales to help you position the field.

You use the Literal Definition screen to edit one map field at a time. After you edit a field (using [PF8] if you need to go to the second page), press PR5. MAPC displays either the Field Definition screen or the Literal Definition screen for the next field that you've selected.

When you've edited all selected fields, MAPC returns you to the Main Menu screen.

Note: Pressing [Enter] on any screen causes it to redisplay so that you can make additional changes if necessary.

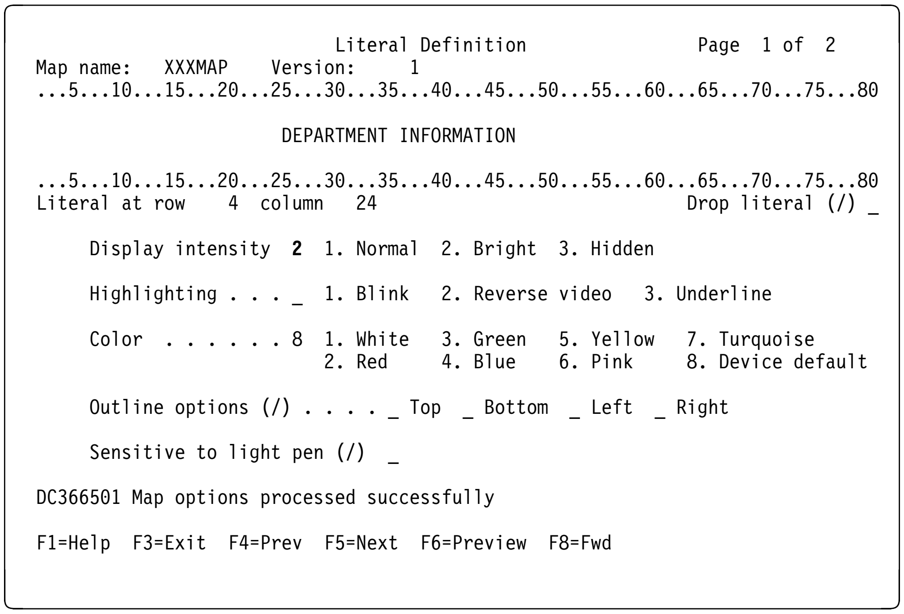
Prompts: In this step, you will use the following prompts on page 1 the Literal Definition screen:

- **Display intensity** — You can specify the runtime display intensity for the literal map field.

  The default, **1 (Normal)**, causes the field to be displayed at normal intensity. You specify **2 (Bright)** to make a field display at bright intensity or **3 (Hidden)** to make a field invisible to the user.

▶▶ For more information on using Literal Definition screen prompts, see the *CA-IDMS Mapping Facility*.

Modifying XXXMAP: Modify the literal field shown on the Literal Definition screen.



```
                              Literal Definition                Page  1 of  2
Map name:   XXXMAP    Version:     1
...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80

                       DEPARTMENT INFORMATION

...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80
Literal at row    4  column   24                             Drop literal (/) _

     Display intensity  2  1. Normal  2. Bright  3. Hidden

     Highlighting . . . _  1. Blink   2. Reverse video   3. Underline

     Color  . . . . . . 8  1. White   3. Green   5. Yellow   7. Turquoise
                           2. Red     4. Blue    6. Pink     8. Device default

     Outline options (/) . . . . _ Top  _ Bottom  _ Left  _ Right

     Sensitive to light pen (/)  _

DC366501 Map options processed successfully

F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F8=Fwd
```

After you press [Enter], MAPC redisplays the Literal Definition screen with a message:

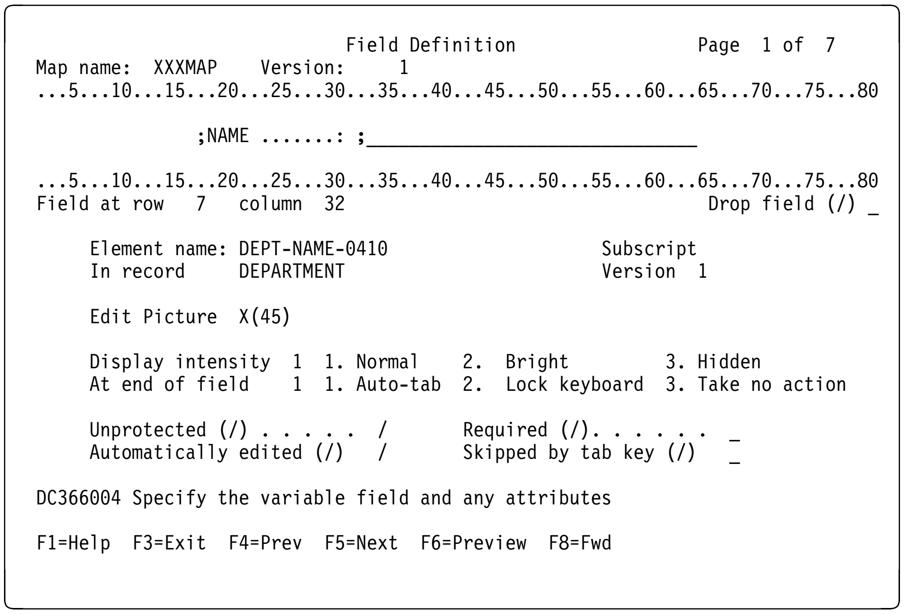
- **Map options process successfully** is returned when the definition contains no errors.
- **An error message** is returned when MAPC detects an error in your definition. In this case, read the message to determine the problem. You can type over any errors and press [Enter] again.

When the current field definition is correct, press [PF5] to see the next definition to be enhanced.

Defining the variable field, DEPT-ID-0410: The next variable field according to the previous table, DEPT-ID-0410, is automatically edited and associated with the auto-tab attribute. These attributes are defaults, and no change needs to be made to this variable field definition on the Field Definition screen.

Defining the variable field, DEPT-NAME-0410: Edit the next variable field according to the previous table. This variable fields requires the auto-tab attribute and a pad character.

Note: The pad character is specified on page 2 of the Field Definition screen.

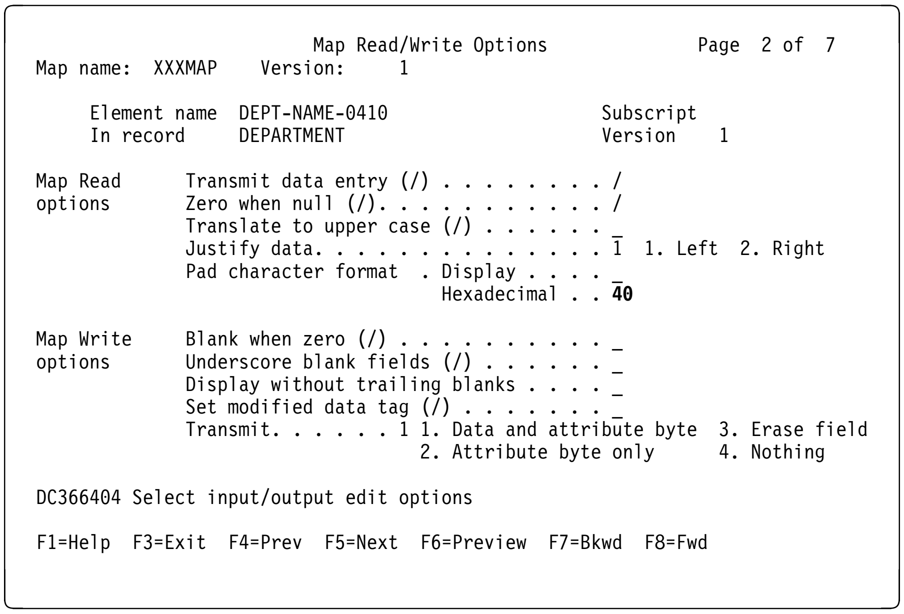


```
                              Field Definition                 Page  1 of  7
Map name:  XXXMAP    Version:     1
...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80

               ;NAME .......: ;_______________________________

...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80
Field at row   7   column  32                               Drop field (/) _

     Element name: DEPT-NAME-0410                  Subscript
     In record     DEPARTMENT                      Version  1

     Edit Picture  X(45)

     Display intensity  1  1. Normal    2.  Bright        3. Hidden
     At end of field    1  1. Auto-tab  2.  Lock keyboard  3. Take no action

     Unprotected (/) . . . . .  /      Required (/). . . . . .  _
     Automatically edited (/)   /      Skipped by tab key (/)   _

DC366004 Specify the variable field and any attributes

F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F8=Fwd
```

Auto-tab is the default.

To define a pad character, you must go to the second page of the Field Definition screen by pressing [PF8].

Sample Field Definition screen — Page 2



```
                          Map Read/Write Options              Page  2 of  7
Map name:  XXXMAP    Version:     1

     Element name  DEPT-NAME-0410                     Subscript
     In record     DEPARTMENT                         Version    1

Map Read      Transmit data entry (/) . . . . . . . . /
options       Zero when null (/). . . . . . . . . . . /
              Translate to upper case (/) . . . . . . _
              Justify data. . . . . . . . . . . . . . 1  1. Left  2. Right
              Pad character format  . Display . . . . _
                                      Hexadecimal . . 40

Map Write     Blank when zero (/) . . . . . . . . . . _
options       Underscore blank fields (/) . . . . . . _
              Display without trailing blanks . . . . _
              Set modified data tag (/) . . . . . . . _
              Transmit. . . . . . 1 1. Data and attribute byte  3. Erase field
                                    2. Attribute byte only      4. Nothing

DC366404 Select input/output edit options

F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F7=Bkwd  F8=Fwd
```

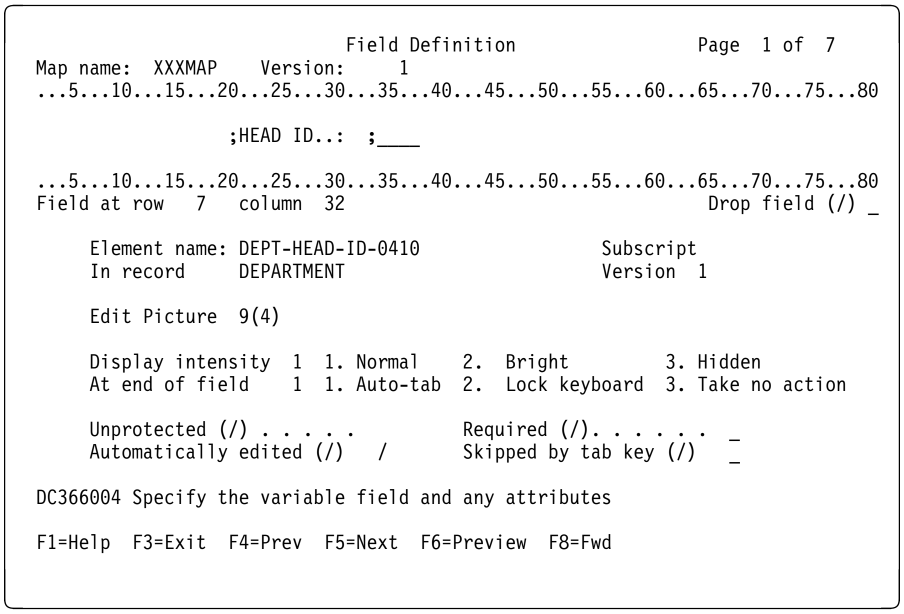
Screen prompts

- **Pad character format** — You can identify a pad character for a map field.

  For map *XXX*MAP, you will assign a pad character to the field that displays department names. Entering "40" (the hexadecimal equivalent of a blank) next to **Hexadecimal** for this field ensures that remaining characters are not stored if the user replaces a long department name (for example, SYSTEMS ENGINEERING DEPARTMENT) with a shorter name (for example, SYSTEMS GROUP) and then clears the rest of the field by pressing the ERASE EOF key.

When the current field definition is correct, press [PF5] to see the next definition to be enhanced.

Defining the variable field, DEPT-HEAD-ID-0410: You can edit the next variable field, DEPT-HEAD-ID-0410, according to the previous table:

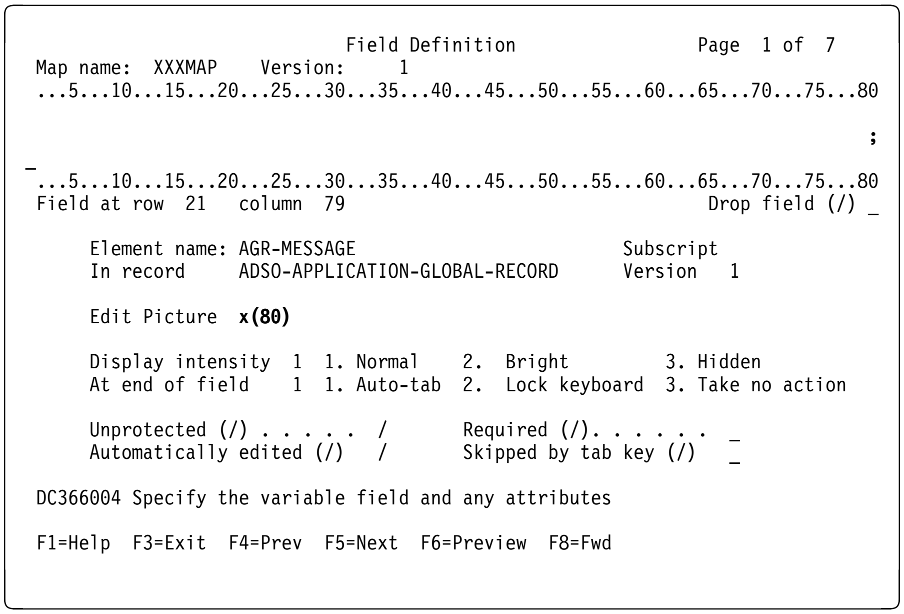


```
                             Field Definition                Page  1 of  7
Map name:  XXXMAP    Version:     1
...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80

                  ;HEAD ID..:  ;____

...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80
Field at row   7   column  32                                 Drop field (/) _

     Element name: DEPT-HEAD-ID-0410                Subscript
     In record     DEPARTMENT                       Version  1

     Edit Picture  9(4)

     Display intensity  1  1. Normal    2.  Bright        3. Hidden
     At end of field    1  1. Auto-tab  2.  Lock keyboard  3. Take no action

     Unprotected (/) . . . . .          Required (/). . . . . .  _
     Automatically edited (/)   /       Skipped by tab key (/)   _

DC366004 Specify the variable field and any attributes

F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F8=Fwd
```

The default is yes for **Automatically edited** and **Auto-tab** for **At end of field**.

When the current field definition is correct, press [PF5] to see the next definition to be enhanced.

Modifying the length of the message field: You can edit the next variable field, the message field, according to the previous table:



```
                             Field Definition                Page  1 of  7
Map name:  XXXMAP    Version:     1
...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80

                                                                              ;
_
 ...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80
Field at row  21   column  79                                 Drop field (/) _

     Element name: AGR-MESSAGE                        Subscript
     In record     ADSO-APPLICATION-GLOBAL-RECORD      Version   1

     Edit Picture  x(80)

     Display intensity  1  1. Normal    2.  Bright        3. Hidden
     At end of field    1  1. Auto-tab  2.  Lock keyboard  3. Take no action

     Unprotected (/) . . . . .  /       Required (/). . . . . .  _
     Automatically edited (/)   /       Skipped by tab key (/)   _

DC366004 Specify the variable field and any attributes

F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F8=Fwd
```

Change the length to 80. The **Edit picture** is automatically updated when you press [Enter].

When you are finished editing fields, you can compile the map, as shown in the next step.

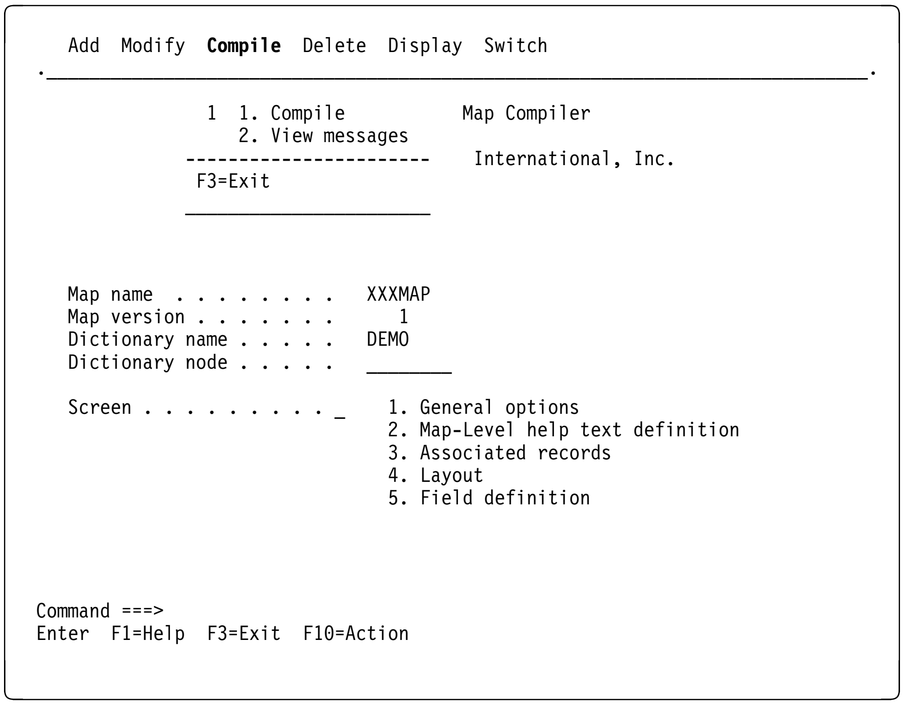
Press [PF5] to return to the Main Menu so that you can compile the map.

8.3.9 Step 9: Compile the map

When you compile a map, MAPC creates a load module that incorporates all of your specifications. You compile an map by selecting the **Compile** activity from the the action bar on the Main Menu screen.

You compile the *XXX*MAP map as shown:

Compiling the map



```
   Add  Modify  Compile  Delete  Display  Switch
 .______________________________________________________________________________.

              1  1. Compile          Map Compiler
                 2. View messages
              -----------------------    International, Inc.
               F3=Exit
              _______________________



   Map name  . . . . . . . .   XXXMAP
   Map version . . . . . . .      1
   Dictionary name . . . . .   DEMO
   Dictionary node . . . . .   ________

   Screen . . . . . . . . . _    1. General options
                                 2. Map-Level help text definition
                                 3. Associated records
                                 4. Layout
                                 5. Field definition




Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

To compile the application, position the cursor on the **Compile** item on the action bar and press [Enter]. You can position the cursor on **Compile** by:

- Tabbing to **Compile** and pressing [Enter]
- Pressing [PF10] to move to the action bar and then tabbing to **Compile** and pressing [Enter]
- Typing **compile** on the command line and pressing [Enter]

Once you have displayed the **Compile** action item, press [Enter] to compile the map.

After you press [Enter] to compile the map, MAPC displays messages to indicate whether the map has been compiled successfully. You will receive a confirming message if the map has compiled successfully.

An error message is displayed when the application cannot be compiled because of an error. In this case, you can display diagnostic information by selecting the **View messages** option on the **Compile** activity on the action bar. Based on this information, you can correct the map and then try to compile the application again.

8.3.10 Exit from MAPC

You can return directly to DC/UCF by pressing [PF3]. Alternatively, you can use the **Switch** activity on the action bar of the Main Menu screen to transfer to another development tool.

In this sample session, you'll exit to DC/UCF so you can execute your application structure. Press [PF3] to exit.

Note: If you suspend MAPC successfully compiling the current map definition, MAPC saves the suspended definition in a queue record associated with your user ID. In an actual production environment, other users will not be able to access the map definition. To enable them to access the definition, specify the **Release** option from the **Modify** activity on the action bar on the Main Menu.

After you exit from MAPC, you can display your map as described below.

8.3.11 Optionally display the map

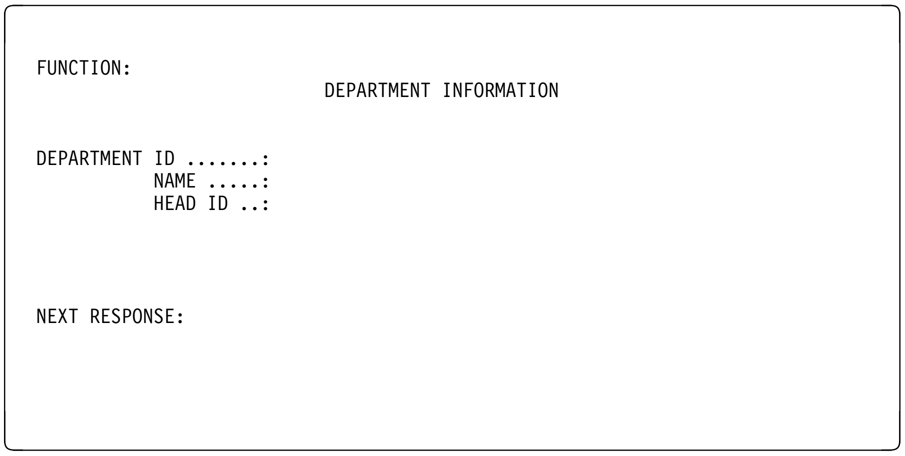
Map *XXX*MAP will be displayed at runtime by dialogs that you define in the next chapter (Chapter 9, “Defining Dialogs Using ADSC” on page 9-1). To display the map before defining any dialogs, do either of the following:

- **In MAPC**, you can display the map on the MAPC Map Image screen. You will do this later in this manual, in Chapter 11, “Modifying a Map Using MAPC” on page 11-1.
- **From DC/UCF**, where you are now, you can display the map by issuing a SHOWMAP command.

Both of the above methods allow you to see how the map will look to an user at runtime. For example, start-field characters are not displayed on the screen; bright fields are displayed in bright intensity on the screen.

Since you have exited from MAPC, you can display map *XXX*MAP directly from DC/UCF:

```
showmap xxxmap
                                                                [Enter]
```



```
FUNCTION:
                           DEPARTMENT INFORMATION


DEPARTMENT ID .......:
           NAME .....:
           HEAD ID ..:




NEXT RESPONSE:
```

Testing the map: While displaying the map, test out how convenient the map is to use. For example:

- Try typing data into unprotected variable fields

  Variable fields on the displayed map do not display or store real data.

- Try pressing the tab key to advance the cursor from field to field.

When you press [Enter], you return to the DC/UCF display.

8.4 Summary

In this chapter, you defined a screen, or **map**, by using MAPC. *XXX*MAP contains two types of fields:

- **Literal fields** — At runtime, literal fields display literal strings.
- **Variable fields** — At runtime, variable fields display stored values and allow users to input values.

You defined the **layout** of fields on the map:

1. **You named the records** that would appear on the map.
2. **You named the elements** of those records that would be displayed.
3. **You used the autopaint facility of MAPC** to create a map automatically.
4. **You modified the placement of the elements and added further literal and variable fields.**
5. **You edited fields** by using the Field Definition and Literal Definition screens. You associated each map variable field, not already associated, with a record element or special system field. You also provided additional field characteristics, such as a pad character.

A map can be used by any number of dialogs. For example, *XXX*MAP is used by dialogs *XXX*DADD and *XXX*DUPD in the sample Department application. You will define these dialogs in Chapter 9, “Defining Dialogs Using ADSC” on page 9-1.

Chapter 9. Defining Dialogs Using ADSC

9.1 Introduction

As the next step in defining an application, you define dialogs by using the CA-ADS dialog compiler (ADSC). The *XXX*DADD and *XXX*DUPD dialogs defined in this chapter are intended for use in the Department application introduced in Chapter 6, “Overview of CA-ADS Application Development” on page 6-1.

This chapter includes:

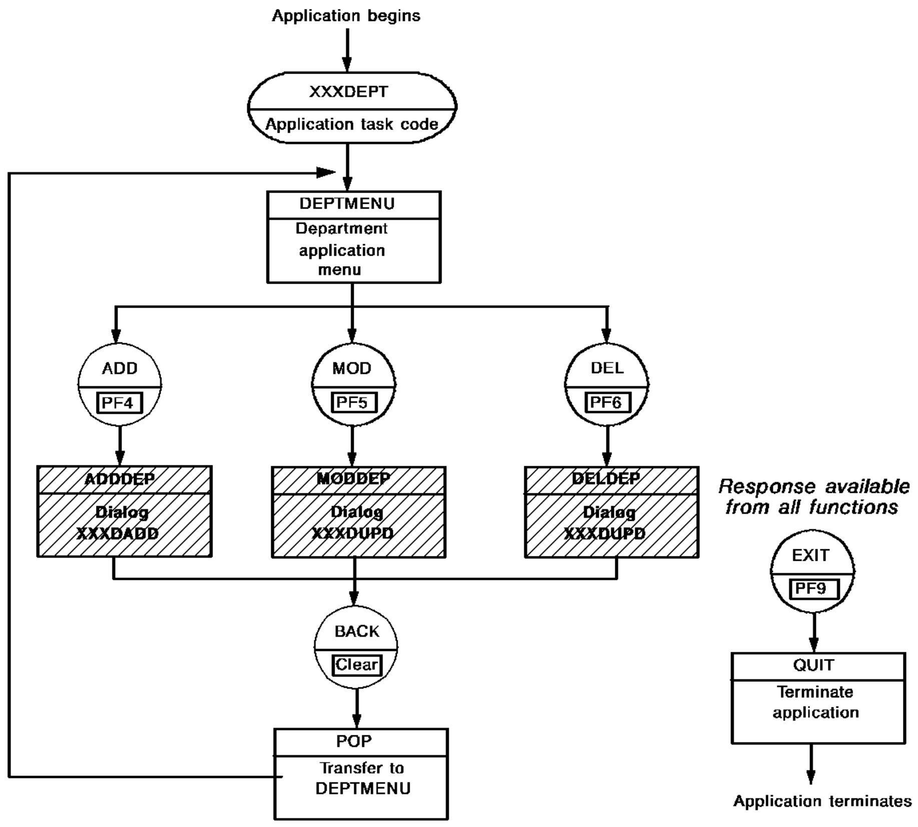
- An overview of developing dialogs for CA-ADS applications
- Instructions for defining the sample *XXX*DADD and *XXX*DUPD dialogs
- Instructions for executing the sample Department application
- A summary of what you've accomplished in this chapter

9.2 Overview

To complete the prototype Department application, you need to define dialogs for the ADDDEP, MODDEP, and DELDEP dialog functions that you created in Chapter 7, “Defining an Application Structure Using ADSA” on page 7-1. When you created ADDDEP, you named *XXX*DADD as the associated dialog. When you created the MODDEP and DELDEP dialog functions, you named *XXX*DUPD as the associated dialog for each function. The *XXX*DADD and *XXX*DUPD dialogs did not exist when you named them in Chapter 7.

In this chapter, you will define the dialogs *XXX*DADD and *XXX*DUPD. The diagram below shows how *XXX*DADD and *XXX*DUPD fit into the Department application. Function ADDDEP invokes dialog *XXX*DADD at runtime. Both MODDEP and DELDEP invoke dialog *XXX*DUPD.

Note: You can substitute your initials for *XXX* in the dialog names.



What is a dialog: A **dialog** is an executable module that consists of components defined by using other development tools. For example, a dialog can include a map defined using MAPC, and modules of process code defined using the IDD menu facility.

At runtime, a dialog:

1. Displays a screen to a user
2. Retrieves entries made by the user

Dialogs in the Department application: For example, in the Department application:

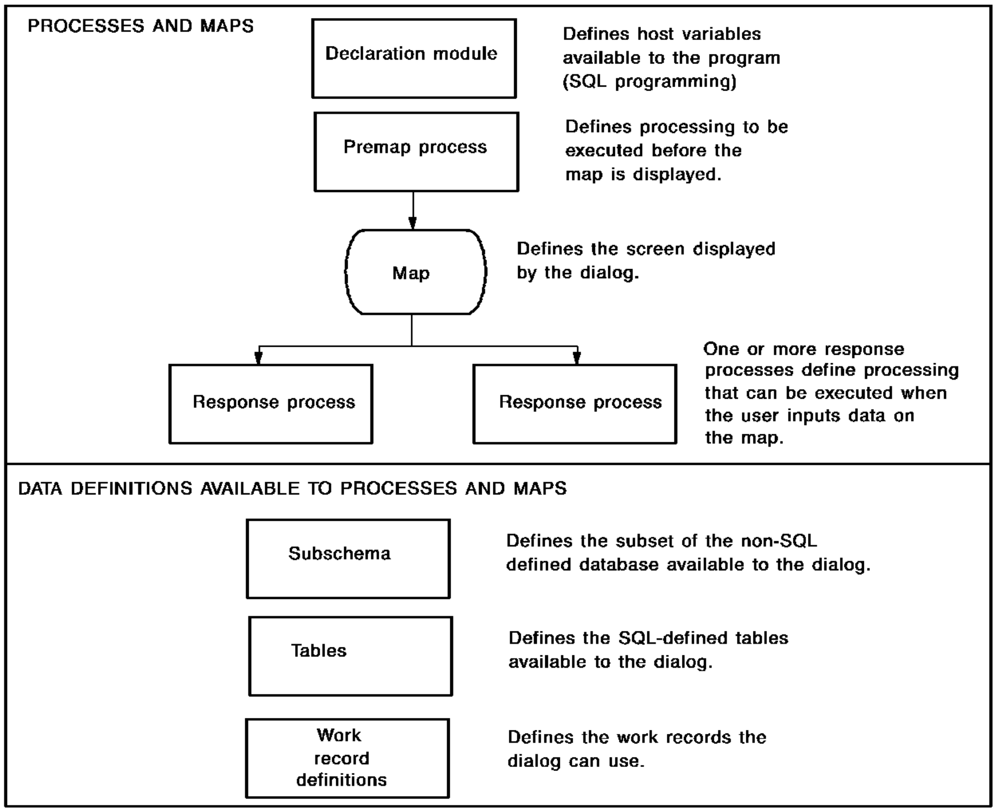
- **Dialog** ***XXX*****DADD** displays a screen that allows users to add department records to the database.
- **Dialog** ***XXX*****DUPD** displays a screen that allows the user to modify or delete existing department records.

Process code can be executed both before the dialog's screen is displayed and after user input is retrieved.

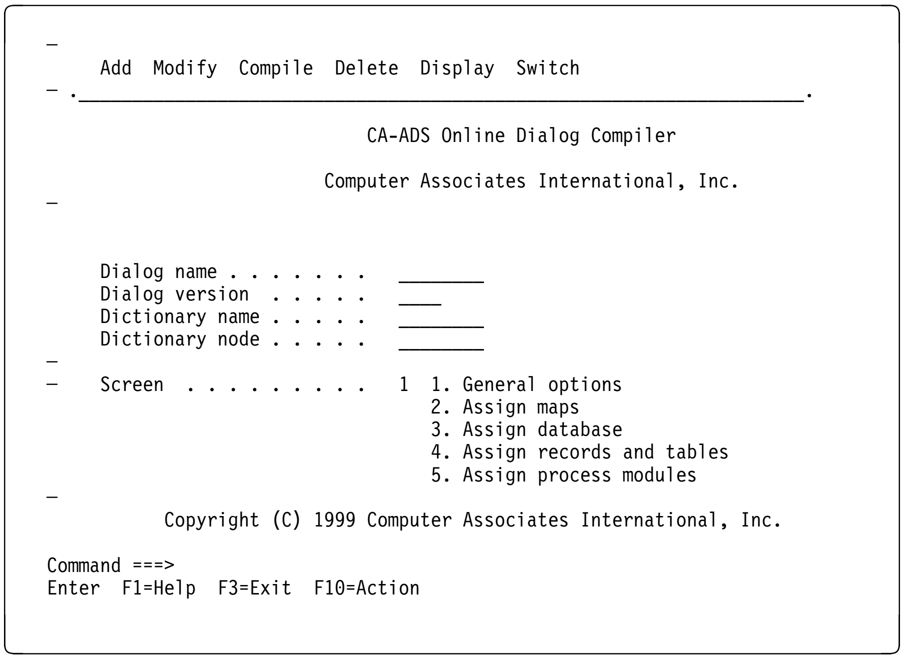
Runtime execution: Dialogs are executed at runtime whenever control passes to the dialog functions with which they are associated. For example, you associated dialog *XXX*DADD with the ADDDEP dialog function when you defined ADDDEP in 7.3.6.1, "Dialog functions" on page 7-24. At runtime, dialog *XXX*DADD will be executed whenever a user invokes the ADDDEP function.

Note: It is possible to create **mapless dialogs** that consist only of process logic. For example, a mapless dialog can be defined to perform database operations. You will not create any mapless dialogs for the sample Department application.

Dialog components: As shown in the diagram, a dialog can consist of several components. When you are developing dialogs for a prototype application, it is only necessary to include maps in the dialogs. Defining basic, skeleton dialogs for a prototype application allows users to execute the application and review the screen displays. Later in the application development cycle, you define process modules that perform processing for the dialogs.


PROCESSES AND MAPS
Declaration module
Defines host variables available to the program (SQL programming)
Premap process
Defines processing to be executed before the map is displayed.
Map
Defines the screen displayed by the dialog.
Response process
Response process
One or more response processes define processing that can be executed when the user inputs data on the map.
DATA DEFINITIONS AVAILABLE TO PROCESSES AND MAPS
Subschema
Defines the subset of the non-SQL defined database available to the dialog.
Tables
Defines the SQL-defined tables available to the dialog.
Work record definitions
Defines the work records the dialog can use.

In this chapter, you will define skeleton dialogs by using the **CA-ADS dialog compiler (ADSC)**. A typical ADSC screen is shown below:



```
_
    Add  Modify  Compile  Delete  Display  Switch
_ .____________________________________________________________________.

                              CA-ADS Online Dialog Compiler

                          Computer Associates International, Inc.
_



    Dialog name . . . . . . .   ________
    Dialog version  . . . . .   ____
    Dictionary name . . . . .   ________
    Dictionary node . . . . .   ________
_
_   Screen  . . . . . . . . .   1  1. General options
                                   2. Assign maps
                                   3. Assign database
                                   4. Assign records and tables
                                   5. Assign process modules
_
          Copyright (C) 1999 Computer Associates International, Inc.

Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

9.3 Instructions for defining dialogs

You use ADSC to define dialogs. To define a skeleton dialog, you invoke ADSC, specify basic information about the dialog, and then create a load module for the dialog.

Steps: In this chapter, you will define skeleton versions of dialogs *XXX*DADD and *XXX*DUPD by using ADSC. You will:

1. Invoke ADSC.
2. Define the dialog *XXX*DADD.
3. Name the associated map.
4. Create a load module for the dialog.
5. Define and compile dialog *XXX*DUPD.

After you compile dialogs *XXX*DADD and *XXX*DUPD, you can exit from ADSC.

Dialogs for the Department application: The following table lists specifications for defining skeleton dialogs *XXX*DADD and *XXX*DUPD. If you need additional information at any time about the use of ADSC, see B.4, “Using ADSC” on page B-16.

Dialog name	Associated map	Purpose of dialog
*XXX*DADD	*XXX*MAP[1]	Allows a user to add a new department record. *XXX*DADD is invoked by the ADDDEP dialog function.
*XXX*DUPD	*XXX*MAP[1]	Allows a user to modify or delete an existing department record. *XXX*DADD is invoked by the MODDEP and DELDEP dialog functions.

[1] You defined map XXXMAP earlier in this sample application development session

9.3.1 Step 1: Invoke ADSC

You can invoke ADSC from CA-IDMS/DC or CA-IDMS/UCF (DC/UCF) by specifying the task code for ADSC (for example, ADSCT) in response to the prompt presented by DC/UCF. For example, you can invoke ADSC from CA-IDMS/DC as shown:

```
ENTER NEXT TASK CODE:
adsct
```

```
Press the ENTER
key to input the --► [Enter]
task code for
ADSC.
```

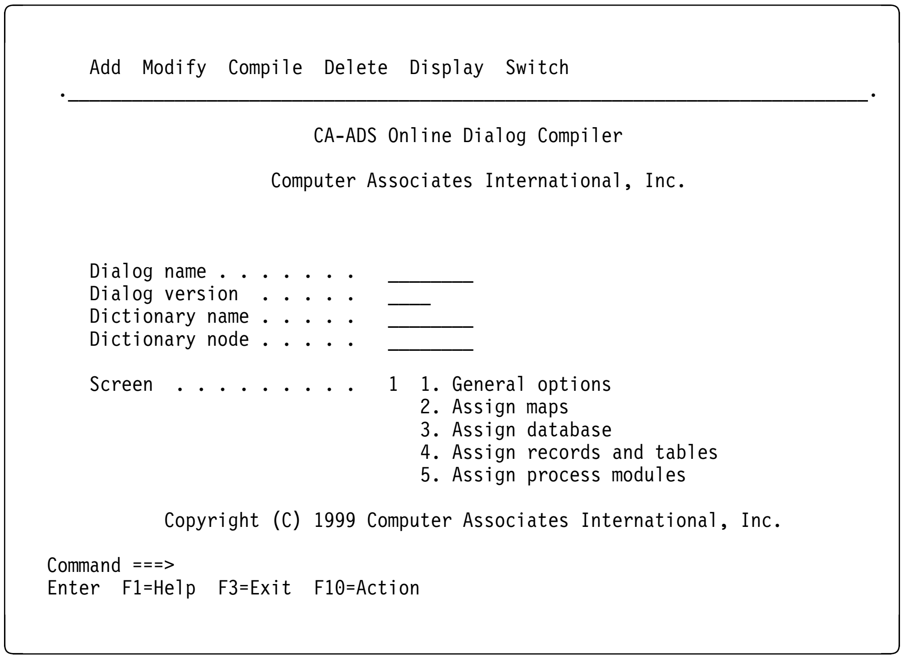
For more information on task codes for CA-ADS development tools, see 6.3, “Application development tools” on page 6-7.

ADSC begins by displaying the Main Menu screen. You define a dialog by using the Main Menu screen as described below.

9.3.2 Step 2: Define dialog XXXDADD

You use the **Main Menu** screen to specify basic information about a dialog. A sample Main Menu screen is shown below:

Sample Main Menu screen



```
    Add  Modify  Compile  Delete  Display  Switch
 .____________________________________________________________________________.
                        CA-ADS Online Dialog Compiler

                    Computer Associates International, Inc.



    Dialog name . . . . . . .   ________
    Dialog version  . . . . .   ____
    Dictionary name . . . . .   ________
    Dictionary node . . . . .   ________

    Screen  . . . . . . . . .   1  1. General options
                                   2. Assign maps
                                   3. Assign database
                                   4. Assign records and tables
                                   5. Assign process modules

           Copyright (C) 1999 Computer Associates International, Inc.

Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

Screen prompts: When you define a skeleton dialog, you typically enter information after one or more of the following Main Menu screen prompts:

- **Dialog name** — You must specify the same dialog name that you specified when you used ADSA to define the associated dialog function.

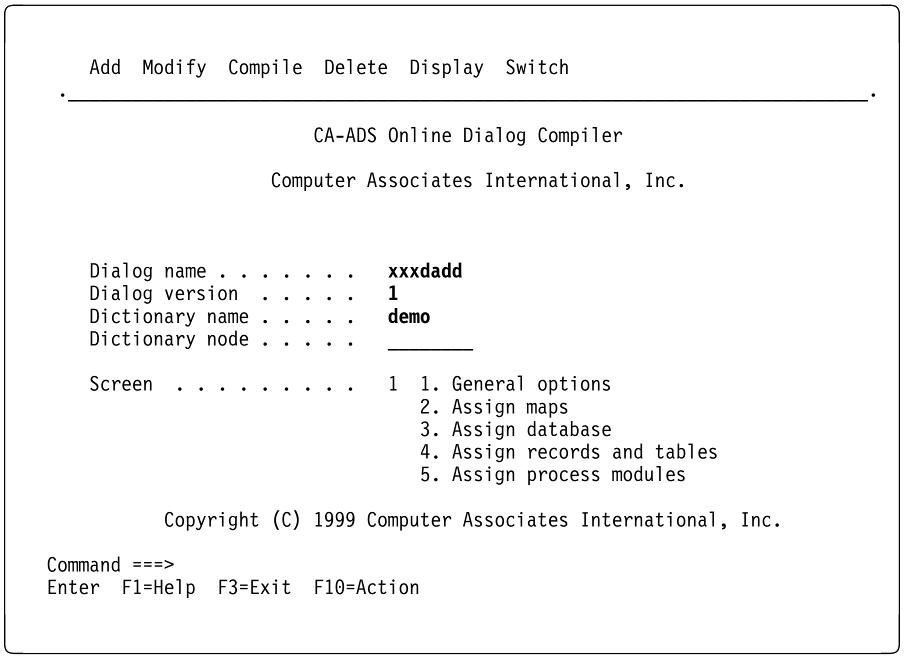
  For example, when you define dialog *XXX*DADD, you must use the same name that you used when you defined the associated ADDDEP function in Chapter 7, “Defining an Application Structure Using ADSA” on page 7-1.

- **Dialog version** — You must specify a version number, in the range 1 through 9999. The default version is 1.

- **Dictionary name** — You must specify the same dictionary (if any) as you specified for your application definition in Chapter 7, “Defining an Application Structure Using ADSA” on page 7-1. The correct dictionary name may already be displayed.

- **Dictionary node** — You must specify the same dictionary node (if any) as you specified for your application definition. The correct dictionary node may already be displayed.

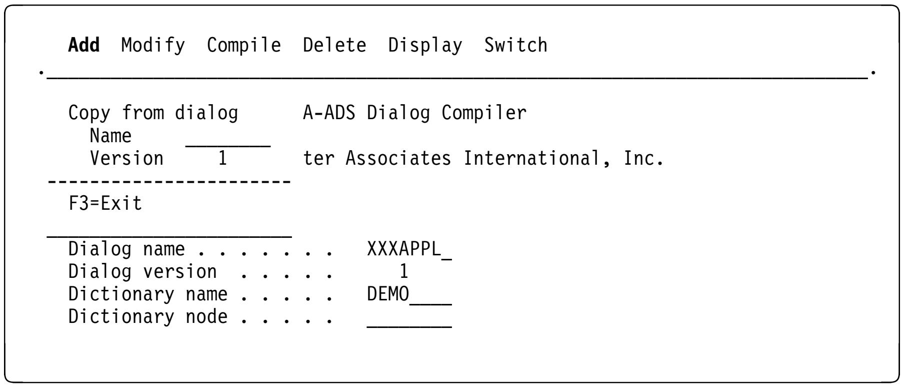
Defining XXXDADD dialog: You can define the *XXX*DADD dialog on the Main Menu screen:



```
    Add  Modify  Compile  Delete  Display  Switch
 .___________________________________________________________________________.

                        CA-ADS Online Dialog Compiler

                    Computer Associates International, Inc.



    Dialog name . . . . . . .   xxxdadd
    Dialog version  . . . . .   1
    Dictionary name . . . . .   demo
    Dictionary node . . . . .   ________

    Screen  . . . . . . . . .   1  1. General options
                                   2. Assign maps
                                   3. Assign database
                                   4. Assign records and tables
                                   5. Assign process modules

           Copyright (C) 1999 Computer Associates International, Inc.

Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

Adding the dialog: To add the dialog, position the cursor on the **Add** item on the action bar and press ENTER. You can position the cursor on **Add** by:

- Tabbing to **Add** and pressing [Enter]
- Pressing [PF10] to move to the action bar and then tabbing to **Add** and pressing [Enter]
- Typing **add** on the command line and pressing [Enter]

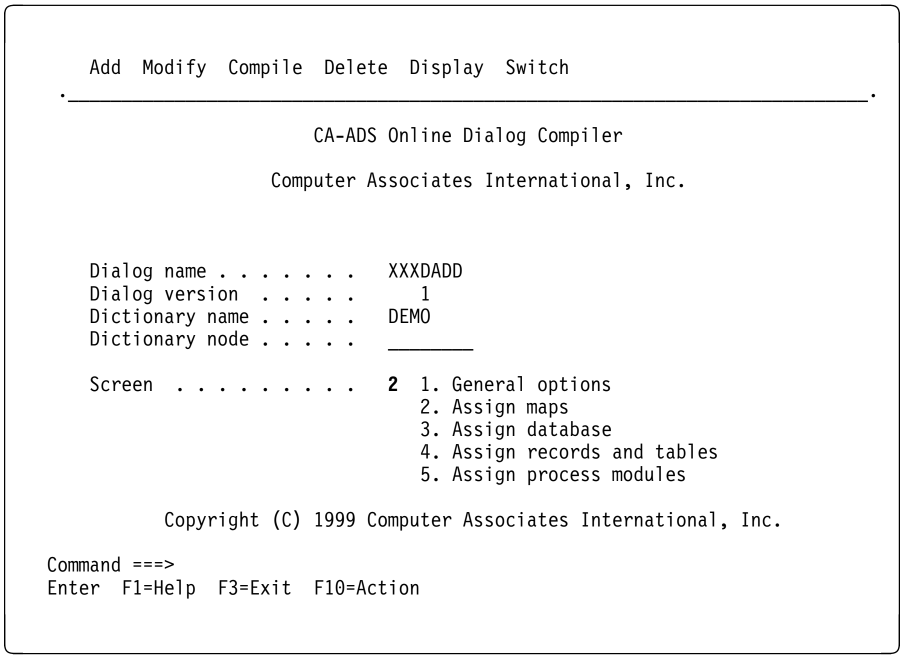


```
  Add  Modify  Compile  Delete  Display  Switch
.______________________________________________________________________________.

  Copy from dialog      A-ADS Dialog Compiler
    Name     ________
    Version     1       ter Associates International, Inc.
-----------------------
  F3=Exit
_______________________
  Dialog name . . . . . . .   XXXAPPL_
  Dialog version  . . . . .      1
  Dictionary name . . . . .   DEMO____
  Dictionary node . . . . .   ________
```

Once you have displayed the **Add** action item, press [Enter] to add the dialog to the dictionary. After you press [Enter], the action is confirmed. If there is an error, an error message is displayed.

9.3.3 Step 3: Name the associated map

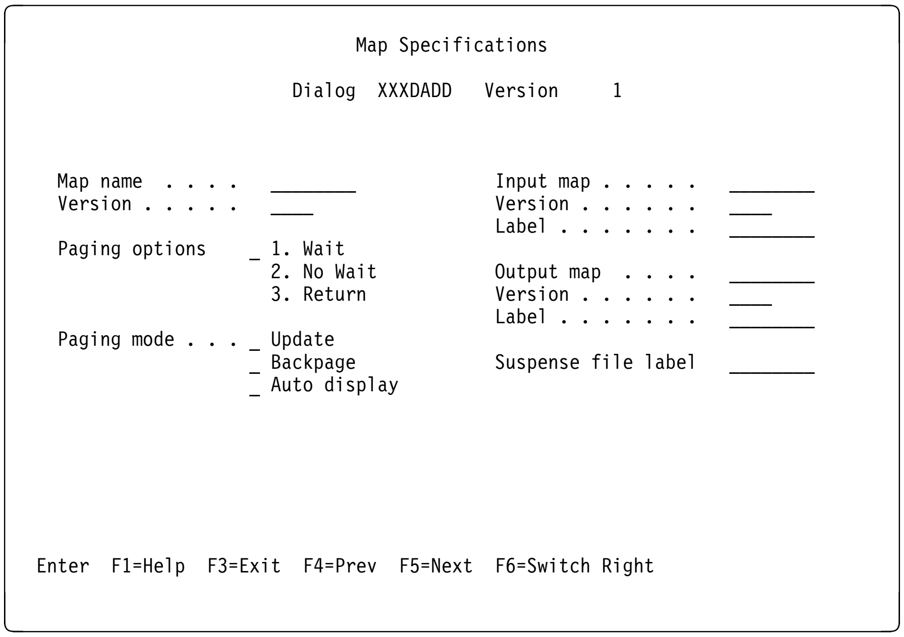
After you specify the name of the dialog, you can name the associated map on the **Map Specifications** screen. You reach the Map Specifications screen by entering **2** next to **Screen** on the Main Menu screen.



```
    Add  Modify  Compile  Delete  Display  Switch
 .____________________________________________________________________________.

                     CA-ADS Online Dialog Compiler

                 Computer Associates International, Inc.



    Dialog name . . . . . . .   XXXDADD
    Dialog version  . . . . .      1
    Dictionary name . . . . .   DEMO
    Dictionary node . . . . .   ________

    Screen  . . . . . . . . .   2  1. General options
                                   2. Assign maps
                                   3. Assign database
                                   4. Assign records and tables
                                   5. Assign process modules

           Copyright (C) 1999 Computer Associates International, Inc.

Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

The following screen is displayed.

Sample Map Specifications screen

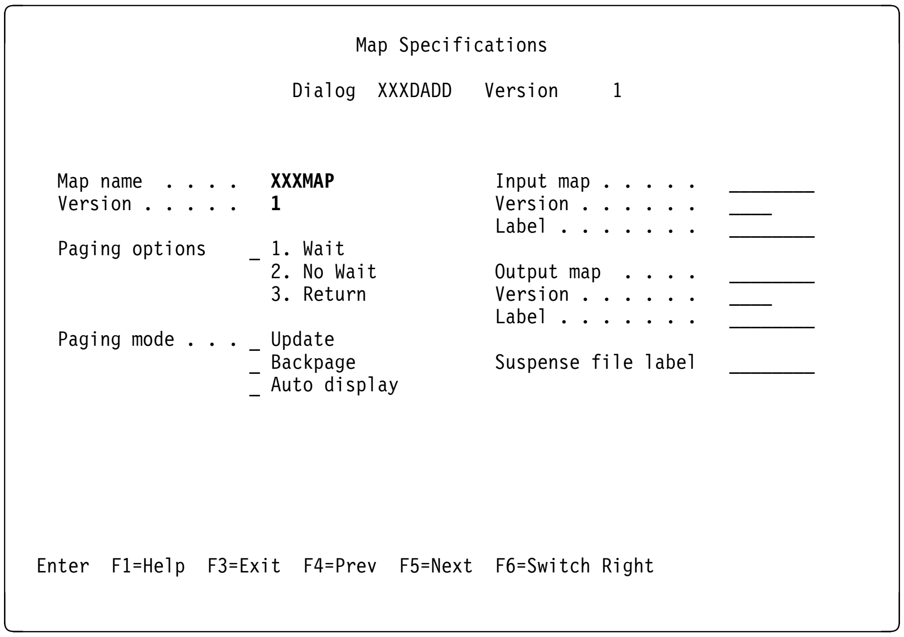


```
                          Map Specifications

                    Dialog  XXXDADD   Version     1



  Map name  . . . .   ________          Input map . . . . .   ________
  Version . . . . .   ____              Version . . . . . .   ____
                                        Label . . . . . . .   ________
  Paging options    _ 1. Wait
                      2. No Wait        Output map  . . . .   ________
                      3. Return         Version . . . . . .   ____
                                        Label . . . . . . .   ________
  Paging mode . . . _ Update
                    _ Backpage          Suspense file label   ________
                    _ Auto display







Enter  F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Switch Right
```

Screen prompts

- **Map name** — You must specify the name of a map created using MAPC. (See Chapter 8, "Defining a Screen Display Using MAPC" on page 8-1.)
- **Map version** — You must specify an existing version of this map.

Associating a map with the dialog



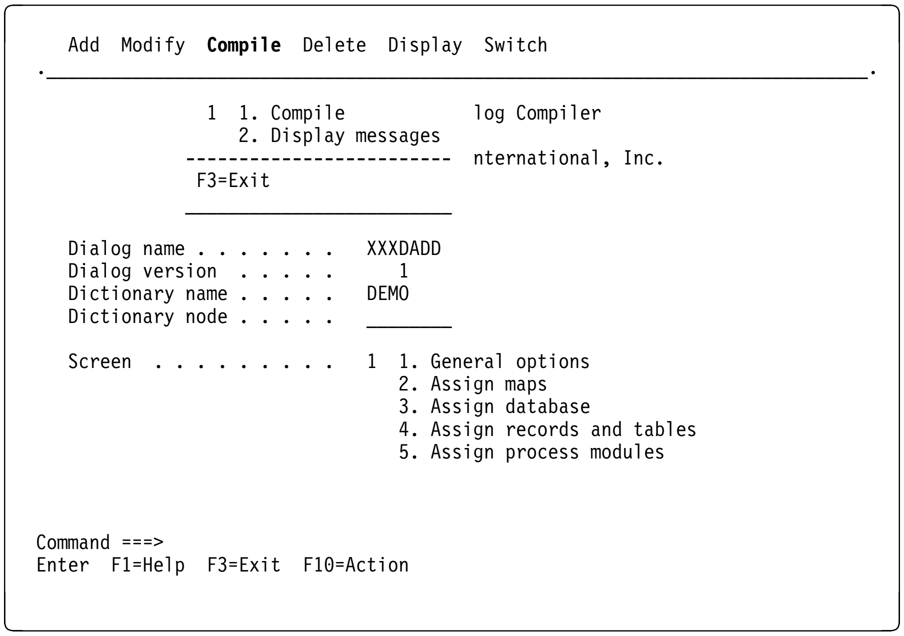
```
                          Map Specifications

                    Dialog  XXXDADD   Version     1



  Map name  . . . .   XXXMAP            Input map . . . . .   ________
  Version . . . . .   1                 Version . . . . . .   ____
                                        Label . . . . . . .   ________
  Paging options    _ 1. Wait
                      2. No Wait        Output map  . . . .   ________
                      3. Return         Version . . . . . .   ____
                                        Label . . . . . . .   ________
  Paging mode . . . _ Update
                    _ Backpage          Suspense file label   ________
                    _ Auto display







Enter  F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Switch Right
```

After you successfully define skeleton dialog *XXX*DADD, you can compile a load module for the dialog as described in the next step. Request the Main Menu by pressing [PF3].

9.3.4 Step 4: Create the XXXDADD dialog load module

When you compile a dialog, ADSC creates a load module that incorporates all of your specifications. You compile a dialog by selecting the **Compile** activity from the the action bar on the Main Menu screen.

Compiling the dialog: You compile the *XXX*DADD dialog as shown:



```
   Add  Modify  Compile  Delete  Display  Switch
 .____________________________________________________________________________.

                1  1. Compile           log Compiler
                   2. Display messages
              -------------------------  nternational, Inc.
               F3=Exit
              _________________________

   Dialog name . . . . . . .   XXXDADD
   Dialog version  . . . . .      1
   Dictionary name . . . . .   DEMO
   Dictionary node . . . . .   ________

   Screen  . . . . . . . . .   1  1. General options
                                  2. Assign maps
                                  3. Assign database
                                  4. Assign records and tables
                                  5. Assign process modules



Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

To compile the application, position the cursor on the **Compile** item on the action bar and press ENTER. You can position the cursor on **Compile** by:

- Tabbing to **Compile** and pressing [Enter]
- Pressing [PF10] to move to the action bar and then tabbing to **Compile** and pressing [Enter]
- Typing **compile** on the command line and pressing [Enter]

Once you have displayed the **Compile** action item, press [Enter] to compile the dialog.

After you press [Enter] to compile the dialog, ADSC displays a message to indicate whether the dialog was compiled successfully.

A confirming message is displayed on the Main Menu screen if the dialog was compiled successfully.

If the dialog could not be compiled, a different message, depending on the nature of the error condition, is displayed. In this case, read the message to determine the problem. After correcting all indicated problems, compile the dialog again.

After you successfully compile the *XXX*DADD dialog, you can define and compile dialog *XXX*DUPD as described below.

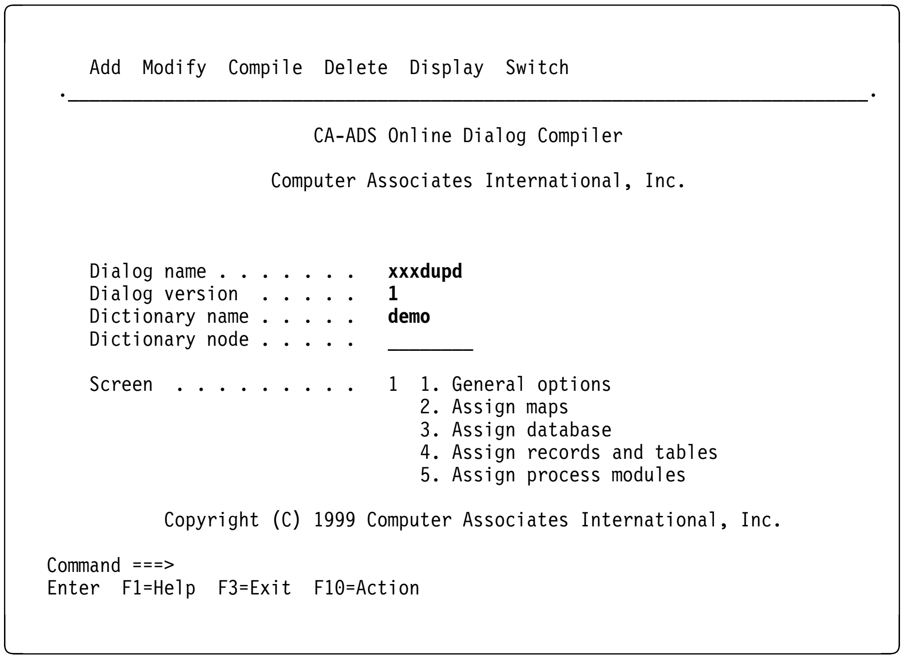
9.3.5 Step 5: Define and compile dialog XXXDUPD

To define *XXX*DUPD, you use the **Main Menu screen**. If you have just compiled dialog *XXX*DADD as described in Step 4, you can define dialog *XXX*DUPD on the Main Menu screen by typing over the dialog name on the screen.

The Department application structure you defined earlier associates dialog *XXX*DUPD with both dialog functions MODDEP and DELDEP. When you define *XXX*DUPD, you must use the same name that you used defining functions MODDEP and DELDEP.

You use the Main Menu screen to define dialog *XXX*DUPD:

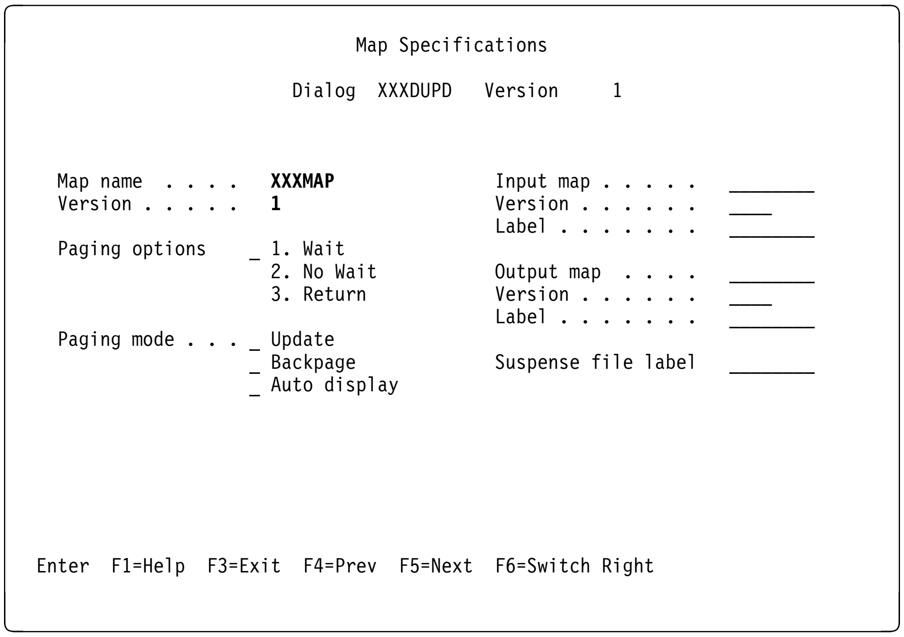
Name the dialog



```
     Add  Modify  Compile  Delete  Display  Switch
  .___________________________________________________________________________.
                           CA-ADS Online Dialog Compiler

                       Computer Associates International, Inc.


     Dialog name . . . . . . .   xxxdupd
     Dialog version  . . . . .   1
     Dictionary name . . . . .   demo
     Dictionary node . . . . .   ________

     Screen  . . . . . . . . .   1  1. General options
                                    2. Assign maps
                                    3. Assign database
                                    4. Assign records and tables
                                    5. Assign process modules

            Copyright (C) 1999 Computer Associates International, Inc.

 Command ===>
 Enter  F1=Help  F3=Exit  F10=Action
```

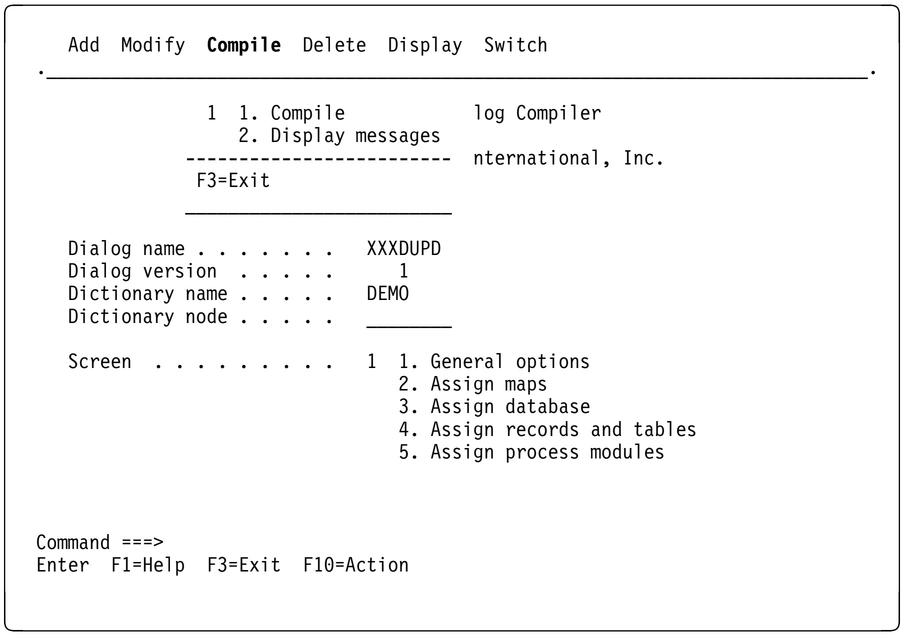
Add the dialog to the dictionary by specifying the **Add** activity from the action bar on the Main Menu. After adding the dialog, choose the **Assign maps** option at the **Screen** prompt to access the **Map Specifications** screen.

Name the map associated with the dialog: Go to the **Map Specifications** screen and name the map. The map is the same map as that associated with the XXXXDADD dialog, XXXMAP.



```
                         Map Specifications

                   Dialog  XXXDUPD   Version     1


  Map name  . . . .   XXXMAP             Input map . . . . .   ________
  Version . . . . .   1                  Version . . . . . .   ____
                                         Label . . . . . . .   ________
  Paging options    _ 1. Wait
                      2. No Wait         Output map  . . . .   ________
                      3. Return          Version . . . . . .   ____
                                         Label . . . . . . .   ________
  Paging mode . . . _ Update
                    _ Backpage           Suspense file label   ________
                    _ Auto display






Enter  F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Switch Right
```

Create the load module: After you define the *XXX*DUPD dialog, you can **create a load module** for the dialog. Return to the Main Menu by pressing [PF3] and select the **Compile** activity.



```
   Add  Modify  Compile  Delete  Display  Switch
.______________________________________________________________________________.

              1  1. Compile           log Compiler
                 2. Display messages
             -------------------------  nternational, Inc.
              F3=Exit
             _________________________

   Dialog name . . . . . . .   XXXDUPD
   Dialog version  . . . . .      1
   Dictionary name . . . . .   DEMO
   Dictionary node . . . . .   ________

   Screen  . . . . . . . . .   1  1. General options
                                  2. Assign maps
                                  3. Assign database
                                  4. Assign records and tables
                                  5. Assign process modules



Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

After you successfully compile the *XXX*DUPD dialog, you can **exit from ADSC** or use the **Switch** activity to access another tool.

9.3.6 Exit from ADSC

In this sample session, you'll exit to DC/UCF so you can execute your application structure again. Press [PF3] to exit from the Main Menu screen to exit.

Note: If you suspend ADSC without successfully compiling the current dialog definition, ADSC saves the suspended definition in a queue record associated with your user ID. In an actual production environment, other users will not be able to access the dialog definition. To enable them to access the definition, specify the **Release** option from the **Modify** activity on the action bar on the Main Menu.

After you exit from ADSC, you can execute your application again as described below.

9.4 Instructions for executing the application

Now that you have defined dialogs for dialog functions ADDDEP, MODDEP, and DELDEP, all menu, system, and dialog functions in the sample Department application are executable.

You now can fully test the prototype Department application. To do this, you will:

1. Invoke the application.
2. Test the application to review screen formats and the flow of control between functions at runtime.
3. Exit from the application when finished.

9.4.1 Step 1: Invoke the application

You execute an CA-ADS application under the CA-ADS runtime system. When you executed the partially defined Department application in Chapter 7, “Defining an Application Structure Using ADSA” on page 7-1, you invoked the application by entering two task codes. The first task code (ADS) invoked the runtime system; the second (*XXX*DEPT) invoked the application itself.

Dynamically associating the task code: To make it easier to execute the application, you can dynamically associate the application's task code with the runtime system. To do this, you issue a DCMT VARY DYNAMIC TASK command while using DC/UCF.

For example, you can enter this DCMT command while using CA-IDMS/DC as shown:

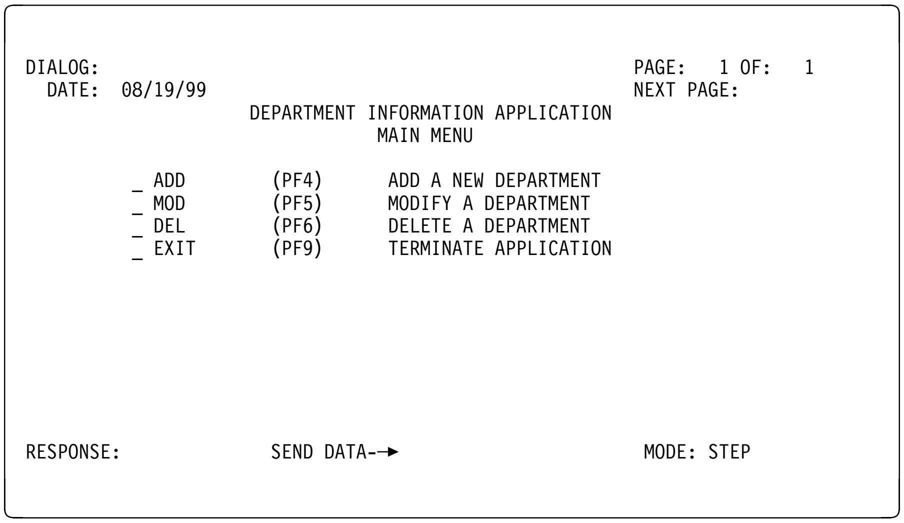
```
ENTER NEXT TASK CODE:
dcmt vary dynamic task xxxdept invokes adsorun1 .
                                                ↑
                                                |
                                           Required space
                                           and period

                                                                    [Enter]
```

ADSORUN1 is the internal name for the CA-ADS runtime system. After you issue the above DCMT command, entering task code *XXX*DEPT invokes the runtime system and then causes the Department application to be executed. This association remains in effect until the system is recycled.

Invoke the application: You can now invoke the Department application from DC/UCF by entering the task code for the application:

```
xxxdept
    ↑
    |
 The Department application's
 task code has been associated
 with the runtime system.
                                                  [Enter]
```

DEPTMENU screen: The DEPTMENU screen is the first screen displayed.



```
DIALOG:                                                PAGE:   1 OF:   1
  DATE:  08/19/99                                      NEXT PAGE:
                    DEPARTMENT INFORMATION APPLICATION
                                MAIN MENU

          _ ADD        (PF4)      ADD A NEW DEPARTMENT
          _ MOD        (PF5)      MODIFY A DEPARTMENT
          _ DEL        (PF6)      DELETE A DEPARTMENT
          _ EXIT       (PF9)      TERMINATE APPLICATION









RESPONSE:              SEND DATA-->                     MODE: STEP
```

After you invoke the Department application from DC/UCF, you can test out features that you've implemented in the prototype Department application, as described below.

9.4.2 Step 2: Test features of the prototype

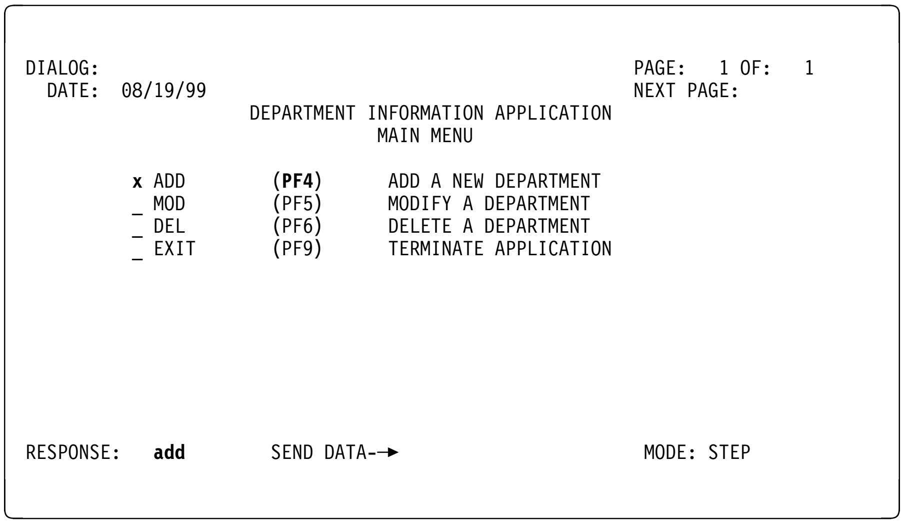
The first function executed in the sample application is DEPTMENU. According to your application design, the following responses are valid from DEPTMENU:

- **ADD** selects dialog function ADDDEP. In the final application, ADDDEP will allow users to add new department information in the database.
- **MOD** selects dialog function MODDEP, which will allow users to modify existing department information in the database.
- **DEL** selects dialog function DELDEP, which will allow users to delete department information.
- **EXIT** selects system function QUIT, which will allow users to leave the application.

Test out each of the above responses while you are executing the Department application prototype. For example, try specifying the ADD response:

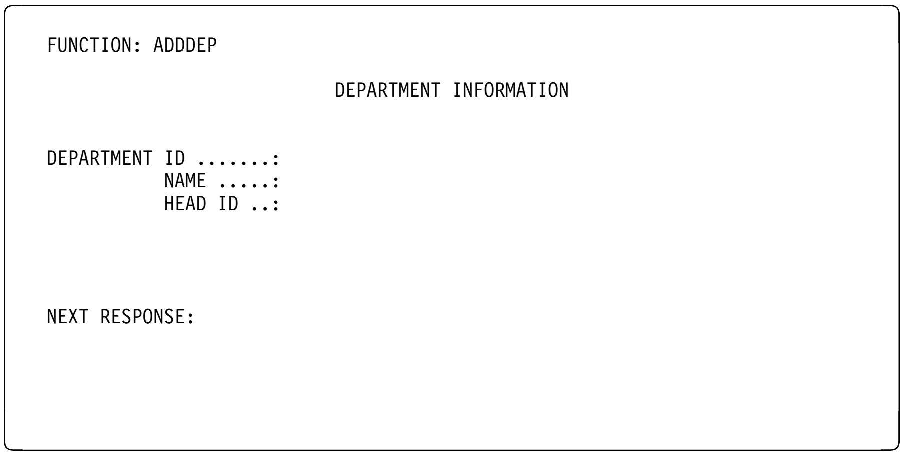
Specifying the ADD response: To specify the ADD response, you can use any of the following methods:

- Press [PF4] to select the ADD response
- Type a nonblank character in front of a response and press [Enter] to select the response.
- Type the response name (ADD) after the RESPONSE: prompt, and press [Enter].



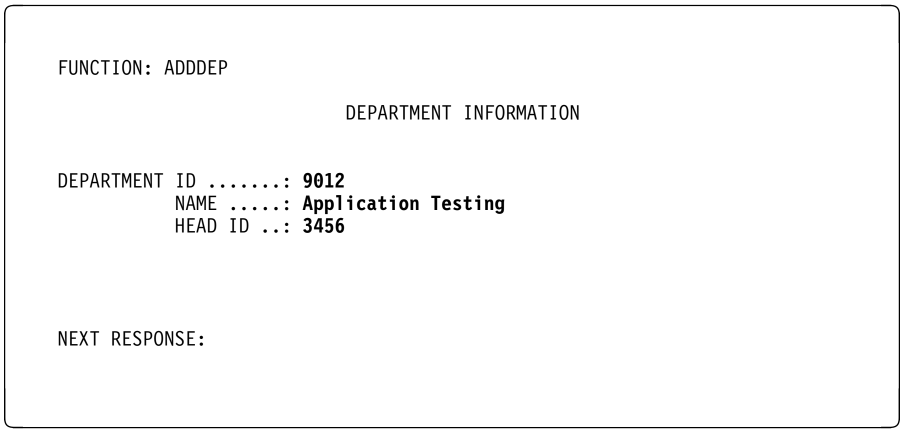
```
DIALOG:                                                 PAGE:   1 OF:   1
  DATE:  08/19/99                                       NEXT PAGE:
                    DEPARTMENT INFORMATION APPLICATION
                                MAIN MENU

          x ADD         (PF4)      ADD A NEW DEPARTMENT
          _ MOD         (PF5)      MODIFY A DEPARTMENT
          _ DEL         (PF6)      DELETE A DEPARTMENT
          _ EXIT        (PF9)      TERMINATE APPLICATION



RESPONSE:   add        SEND DATA-->                      MODE: STEP
```

ADDDEP function: The ADDDEP function is displayed with the *XXX*MAP.



```
  FUNCTION: ADDDEP

                           DEPARTMENT INFORMATION


  DEPARTMENT ID .......:
             NAME .....:
             HEAD ID ..:



  NEXT RESPONSE:
```

The ADDDEP function allows the user to enter information about a department. Try tabbing between variable fields and entering sample department information. You cannot type anything in the FUNCTION: field because you made it a protected field.



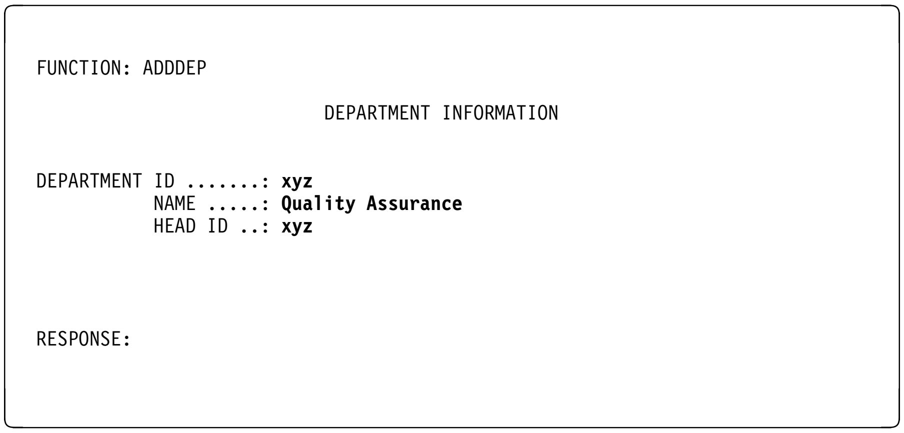
```
FUNCTION: ADDDEP
                           DEPARTMENT INFORMATION


DEPARTMENT ID .......: 9012
           NAME .....: Application Testing
           HEAD ID ..: 3456




NEXT RESPONSE:
```

ADDDEP is a skeleton dialog, which means that you haven't added any process logic to the dialog. The dialog cannot access the database. Therefore, your sample data is not stored when you press [Enter].

Input-handling operations: Other input-handling operations are performed automatically at runtime. For example, your sample input is:

- **Echoed on the screen** after you press [Enter].
- **Tested for invalid values** and redisplayed in bold when errors are found.

  For example, try entering invalid values in the DEPARTMENT ID and HEAD ID variable fields:



```
FUNCTION: ADDDEP
                           DEPARTMENT INFORMATION


DEPARTMENT ID .......: xyz
           NAME .....: Quality Assurance
           HEAD ID ..: xyz




RESPONSE:
```

In this case, the runtime system returns an error message:

```
**ERROR AT 7,24** **ERROR AT 9,24**
```

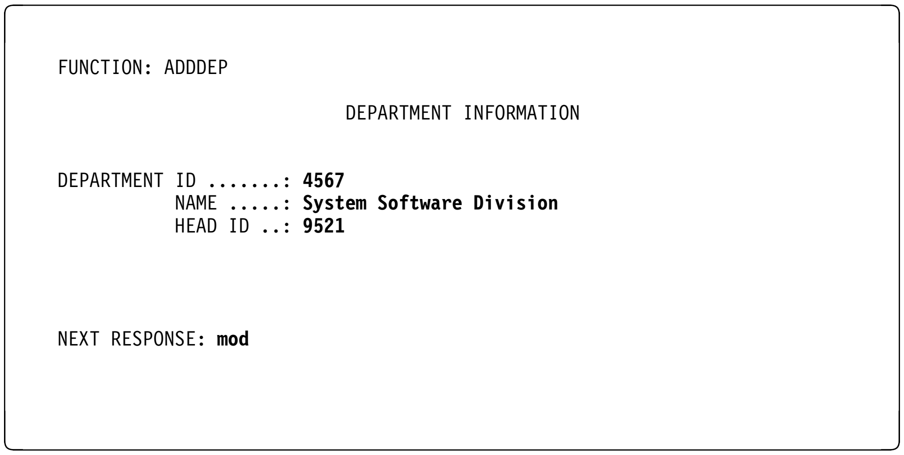
Nonnumeric data is invalid for the DEPARTMENT ID and HEAD ID variable fields because you enabled the CA-ADS automatic editing feature for these numeric fields in in Chapter 8.

▸▸ For more information on how automatic editing can be used to keep users from entering invalid values, see the *CA-IDMS Mapping Facility*.

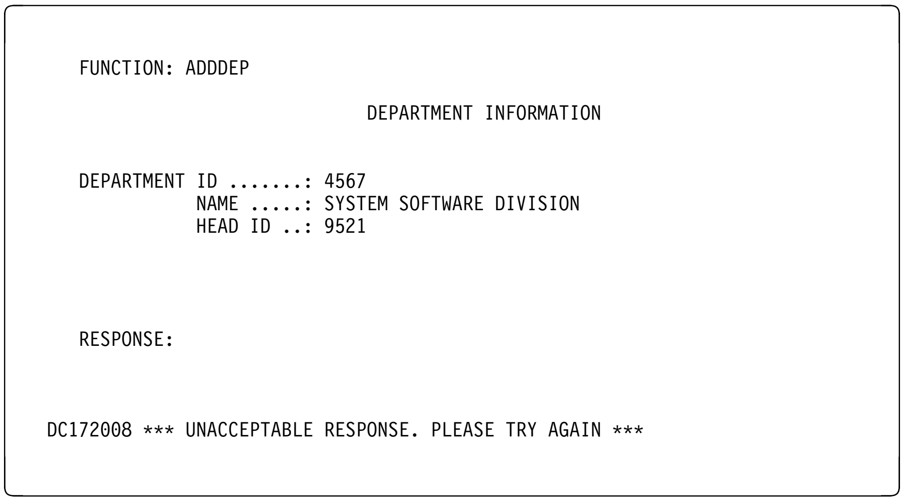
You also can test the NEXT RESPONSE variable field. You can try entering undefined responses (for example, UPDATE) or responses that are valid for the application but not for the ADDDEP function (for example, MOD). In fact, a user testing the Department application prototype probably would try to access the MODDEP function from ADDDEP to see if newly added department information can be modified easily if, for example, the department name is misspelled.

Enter new department values on the screen, and then specify the MOD response to try accessing MODDEP from ADDDEP:

Specifying the MOD response from ADDDEP



```
 FUNCTION: ADDDEP
                            DEPARTMENT INFORMATION


 DEPARTMENT ID .......: 4567
            NAME .....: System Software Division
            HEAD ID ..: 9521




 NEXT RESPONSE: mod
```

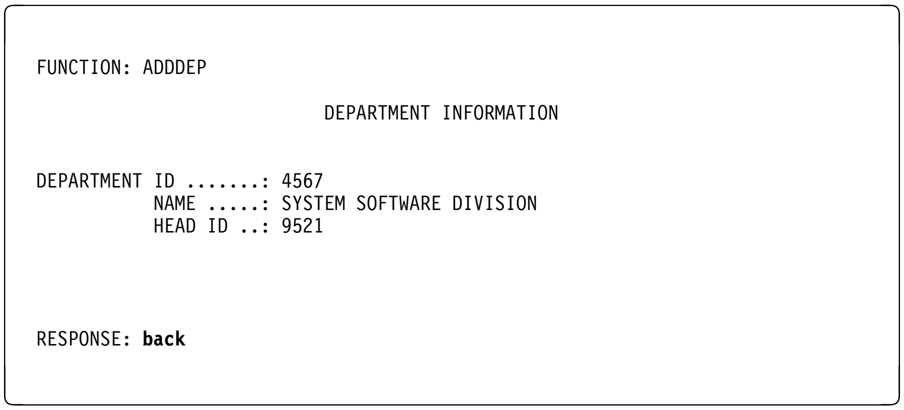


```
   FUNCTION: ADDDEP
                              DEPARTMENT INFORMATION


   DEPARTMENT ID .......: 4567
              NAME .....: SYSTEM SOFTWARE DIVISION
              HEAD ID ..: 9521




   RESPONSE:



DC172008 *** UNACCEPTABLE RESPONSE. PLEASE TRY AGAIN ***
```

Even though MOD is defined for the application, it is not valid from the ADDDEP function. According to your application definition, only the following responses are valid from ADDDEP:

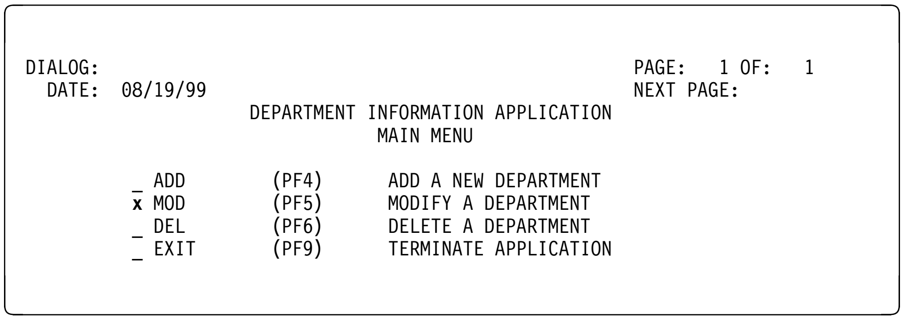
- **BACK** selects system function POP, which returns execution to the previous menu function (in this case, DEPTMENU).
- **EXIT** selects system function QUIT, which terminates the application.

Display the MODDEP function: To display the MODDEP function from ADDDEP, you must first access a function from which MODDEP is valid. Since MOD is valid for the DEPTMENU function, use the **BACK** response to return to DEPTMENU, and then invoke MOD from DEPTMENU:



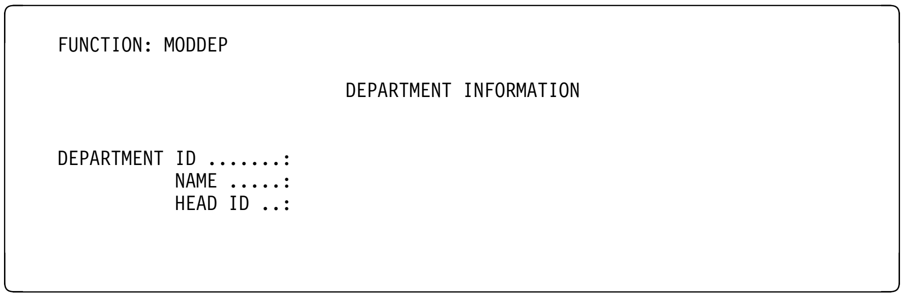
```
FUNCTION: ADDDEP
                           DEPARTMENT INFORMATION


DEPARTMENT ID .......: 4567
           NAME .....: SYSTEM SOFTWARE DIVISION
           HEAD ID ..: 9521




RESPONSE: back
```

Choose MOD from the DEPTMENU screen to display the MODDEP function.



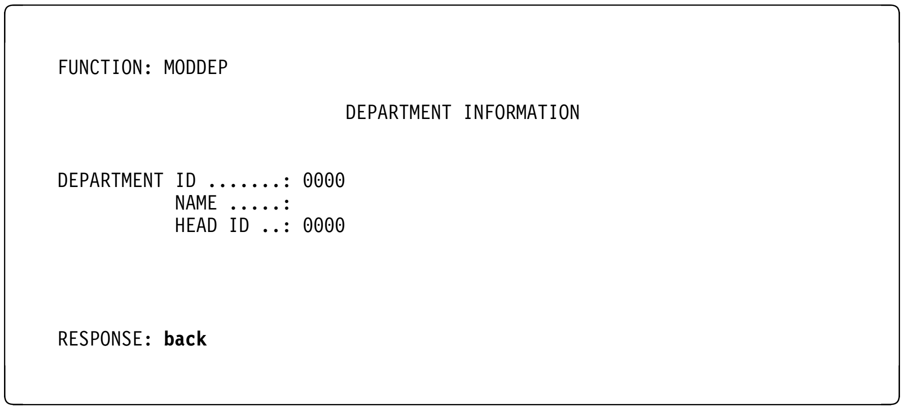
```
DIALOG:                                                 PAGE:   1 OF:   1
  DATE:  08/19/99                                       NEXT PAGE:
                    DEPARTMENT INFORMATION APPLICATION
                                MAIN MENU

          _ ADD        (PF4)      ADD A NEW DEPARTMENT
          x MOD        (PF5)      MODIFY A DEPARTMENT
          _ DEL        (PF6)      DELETE A DEPARTMENT
          _ EXIT       (PF9)      TERMINATE APPLICATION
```

The MODDEP function is displayed



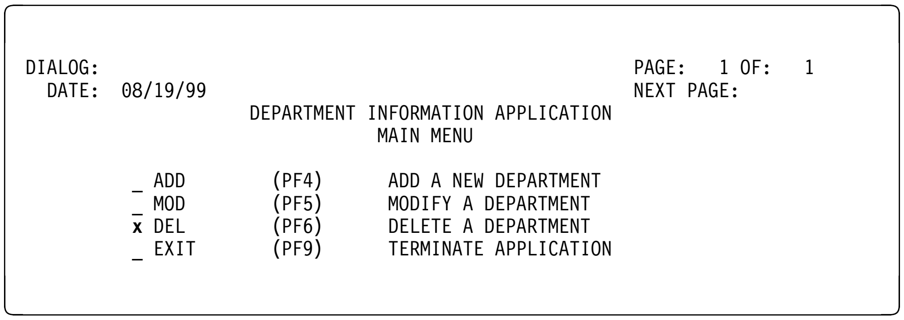
```
  FUNCTION: MODDEP

                             DEPARTMENT INFORMATION


  DEPARTMENT ID .......:
             NAME .....:
             HEAD ID ..:
```

Display the DELDEP function: To display function DELDEP, you first return to the DEPTMENU function. From DEPTMENU, you can invoke DELDEP as shown:



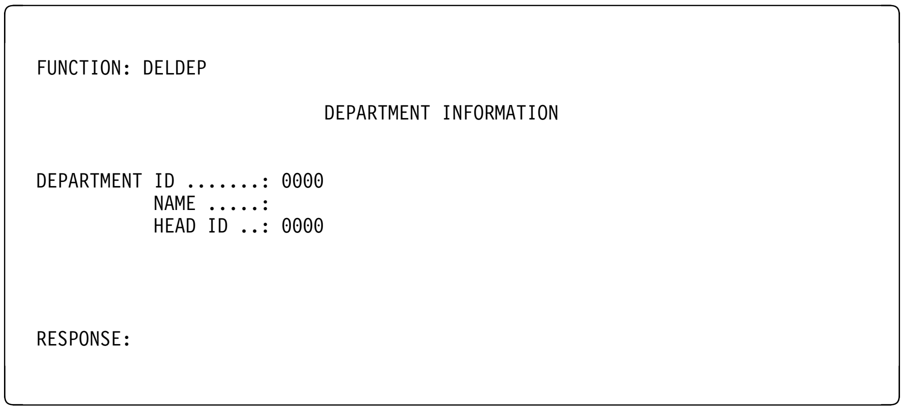
```
  FUNCTION: MODDEP

                             DEPARTMENT INFORMATION


  DEPARTMENT ID .......: 0000
             NAME .....:
             HEAD ID ..: 0000




  RESPONSE: back
```

Select the DEL response to display DELDEP



```
DIALOG:                                           PAGE:   1 OF:   1
  DATE:  08/19/99                                 NEXT PAGE:
                    DEPARTMENT INFORMATION APPLICATION
                                MAIN MENU

           _ ADD        (PF4)      ADD A NEW DEPARTMENT
           _ MOD        (PF5)      MODIFY A DEPARTMENT
           x DEL        (PF6)      DELETE A DEPARTMENT
           _ EXIT       (PF9)      TERMINATE APPLICATION
```

The DELDEP function is displayed.



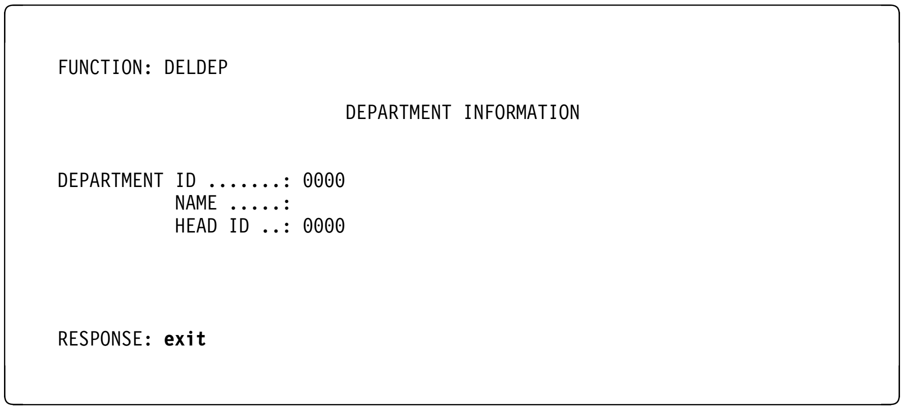
```
FUNCTION: DELDEP
                           DEPARTMENT INFORMATION


DEPARTMENT ID .......: 0000
           NAME .....:
           HEAD ID ..: 0000




RESPONSE:
```

Continue to test the application prototype until you are familiar with the Department application prototype. When you are finished testing the application, exit from the application as described below.

9.4.3 Step 3: Exit from the application

When you are ready to exit from the application, select the EXIT response. The EXIT response invokes the QUIT system response, which terminates the application and returns control to DC/UCF.

When you defined the Department application structure, you defined EXIT to be available from all functions in the application. You can exit from the Department application by selecting EXIT from any function and pressing the [Enter] key. For example, while using the DELDEP function, you can exit from the Department application as shown:



```
FUNCTION: DELDEP
                           DEPARTMENT INFORMATION


DEPARTMENT ID .......: 0000
           NAME .....:
           HEAD ID ..: 0000




RESPONSE: exit
```

```
You can also press ───► [PF9]
the control key
associated with EXIT.
```

If you want to execute the prototype again, specify the application's task code to DC/UCF and press [Enter], as described above in 9.4.1, “Step 1: Invoke the application” on page 9-17.

9.5 Summary

Developers and users can execute a preliminary (that is, **prototype**) application early in the development cycle. The prototype can be used to test the user interface of the application and to provide a milestone in the application development cycle. In this chapter, you completed and executed the prototype of the Department application. By executing the prototype, you were able see how a user might use the application.

You created the Department application prototype by defining application components as described below:

1. **You defined the application structure** in Chapter 7, "Defining an Application Structure Using ADSA" on page 7-1. The application structure consists of:
   - **Functions**, which represent the units of work to be performed by the application. You defined menu, system, and dialog functions for the Department application.
   - **Responses**, which establish runtime paths between the application functions.
   - **Task codes**, which establish entry points into the application.
2. **You defined a map**, or screen display, in Chapter 8, "Defining a Screen Display Using MAPC" on page 8-1. The map you created will be displayed by the dialogs in the Department application.
3. **You defined skeleton dialogs** for the ADDDEP, MODDEP, and DELDEP dialog functions in this chapter. Defining skeleton dialogs allowed you to execute the application and test flow of control and screen displays. In later chapters, you will create modules of process code for the ADDDEP, MODDEP, and DELDEP dialogs.

Changes to the application: Based on tests made using the prototype, users and developers often suggest modifications to the application. For example, users who test out the Department application probably would request that a path be defined that leads directly from ADDDEP to the MODDEP function.

Other changes to the application may be suggested to make the application conform to site conventions. For example, one convention is to use the [PF3] to leave an application Both of the above changes to the Department application can be made by using the application compiler (ADSA), as detailed in the next chapter.

Users also can suggest changes to maps. For example, users might request that key data be displayed in bright intensity and that error messages for variable fields provide specific information. MAPC is used to make changes to maps, as described in Chapter 11, "Modifying a Map Using MAPC" on page 11-1.

Chapter 10. Modifying the Application Structure Using ADSA

10.1 Introduction

Developers and end users can execute the prototype of an CA-ADS application to review the application's structure and user interface. Based on the prototype, they can suggest changes to the application. This chapter provides instructions for modifying the Department application structure.

This chapter includes:

- An overview of modifying an application structure in the CA-ADS environment
- Instructions for modifying the sample Department application
- A summary of what you've accomplished in this chapter

10.2 Overview

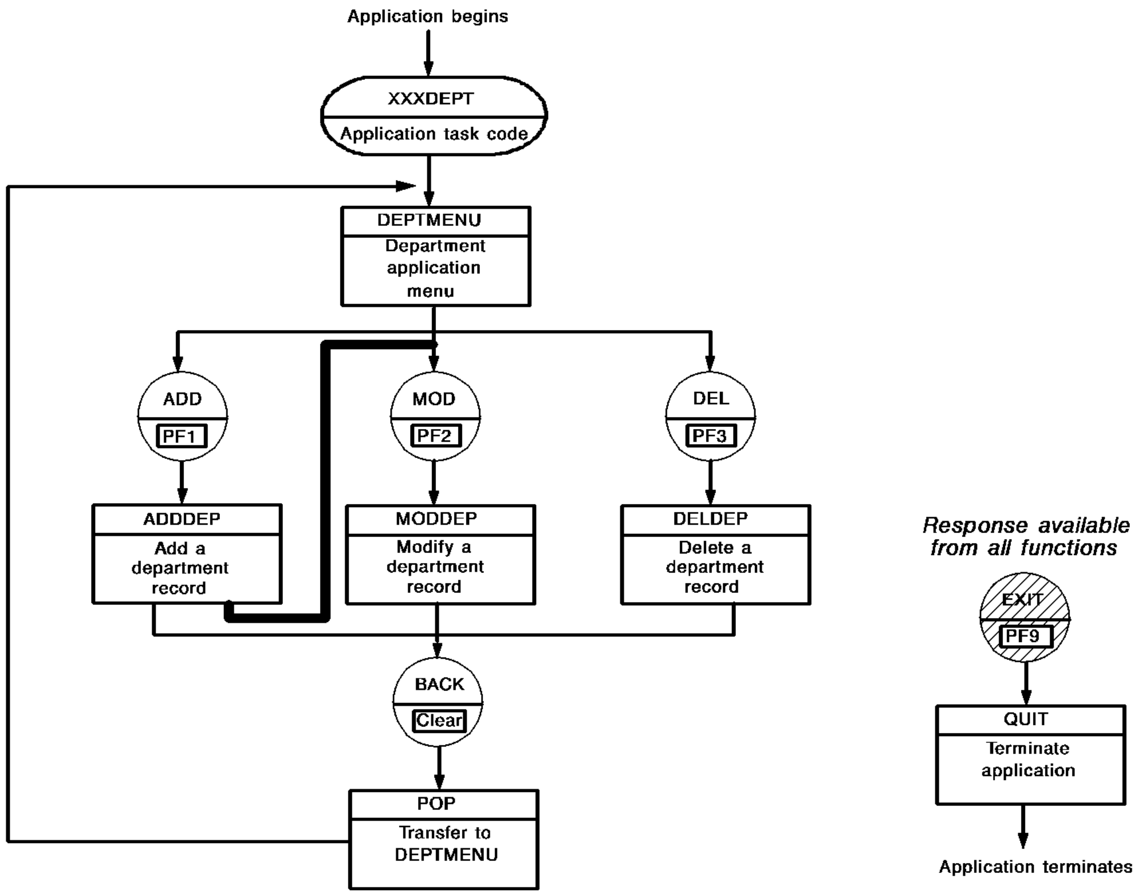
User requests and design alterations can be incorporated easily into a prototype application. Minor changes, such as changing the function key that invokes a response, and major changes, such as adding new responses and functions to the application, can be performed quickly and easily.

You modify application components by using the same tools you use to define the components. To modify the Department application structure, you will use the **CA-ADS application compiler (ADSA)**.

In this chapter, you will use ADSA to:

1. **Modify the EXIT response** so that [PF3] invokes the response
2. **Modify the ADDDEP function** so that the MOD response is valid directly from ADDDEP

The following diagram shows how the modifications affect the structure of the Department application. Using ADSA, you will make the MOD response valid from the ADDDEP function and also change the function key for the EXIT response. Instructions for modifying the sample application and executing the modified application are provided on the following pages.


Application begins
XXXDEPT
Application task code
DEPTMENU
Department application menu
ADD
PF1
MOD
PF2
DEL
PF3
ADDDEP
Add a department record
MODDEP
Modify a department record
DELDEP
Delete a department record
BACK
Clear
POP
Transfer to DEPTMENU
Response available from all functions
EXIT
PF9
QUIT
Terminate application
Application terminates

10.3 Instructions

To modify the Department application, you will perform the following steps:

1. Retrieve the Department application.
2. Select responses and functions to modify.
3. Modify the EXIT response.
4. Modify the ADDDEP function.
5. Create a load module for the modified application.

After you modify the application, you can exit from ADSA and execute the application to see how your changes impact runtime flow of control. If you need additional information at any time about the use of ADSA, see Chapter 7, “Defining an Application Structure Using ADSA” on page 7-1.

10.3.1 Step 1: Retrieve the application to be modified

Invoking ADSA: In order to modify an application, you must first invoke ADSA and then use ADSA to retrieve the application definition.

You invoke ADSA from CA-IDMS/DC or CA-IDMS/UCF (DC/UCF) by specifying the task code for ADSA (for example, ADSAT) in response to the prompt presented by DC/UCF. For example, when using CA-IDMS/DC, you invoke ADSA as shown:

```
ENTER NEXT TASK CODE:
adsat

                                                     [Enter]
```

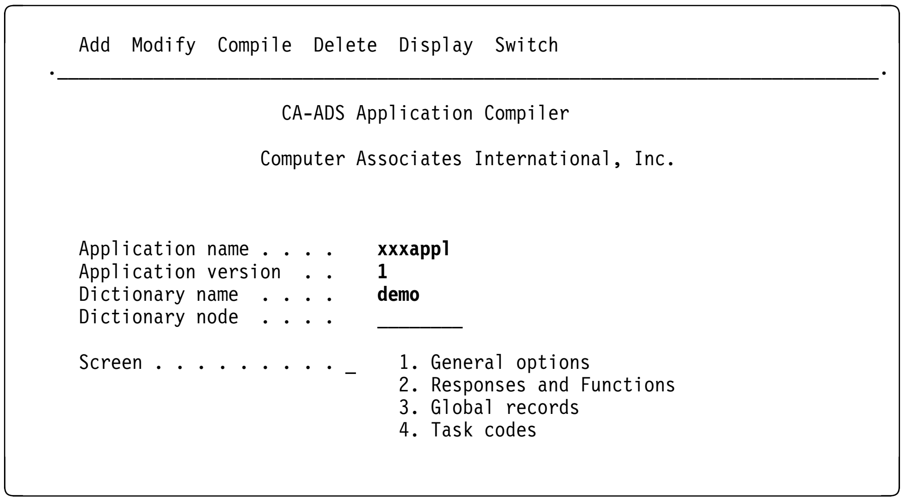
For more information on invoking ADSA, see 7.3.1, “Step 1: Invoke ADSA” on page 7-9.

ADSA begins by displaying the **Main Menu** screen. You use the ADSA Main Menu screen to retrieve an application definition for modification. To retrieve an application, you typically enter information after one or more of the following Main Menu screen prompts:

Screen prompts

- **Application name** — You must specify the name (for example, *XXX*DEPT) that you used when you defined the application in Chapter 7.
- **Dictionary name** — You must specify the same dictionary, if any, as you specified for your application definition in Chapter 7. The correct dictionary name may already be displayed in this field.
- **Dictionary node** — You must specify the same dictionary node, if any, as you specified for your application definition in Chapter 7. The correct dictionary node may already be displayed in this field.

Retrieving the Department application: Use the ADSA Main Menu screen to retrieve the Department application:



```
   Add  Modify  Compile  Delete  Display  Switch
 .____________________________________________________________________________.

                     CA-ADS Application Compiler

                   Computer Associates International, Inc.



   Application name . . . .    xxxappl
   Application version  . .    1
   Dictionary name  . . . .    demo
   Dictionary node  . . . .    ________

   Screen . . . . . . . . . _    1. General options
                                 2. Responses and Functions
                                 3. Global records
                                 4. Task codes
```

After you press [Enter], ADSA redisplays the Main Menu screen with a message confirming that the application is available for modification.

Note: If the application has not been explicitly released (using the **Release** option of the **Modify** action on the action bar of the Main Menu), naming the application on the Main Menu screen retrieves that definition for modification. If the application has been released, you use the ADSA Main Menu screen to check out the application definition for modification (using the **Checkout** option of the **Modify** action on the action bar of the Main Menu).

For information on checking out an application, see *CA-ADS Reference*.

If the application has been released, subsequently checked out to another developer and not released by that developer, you will not be able to check it out.

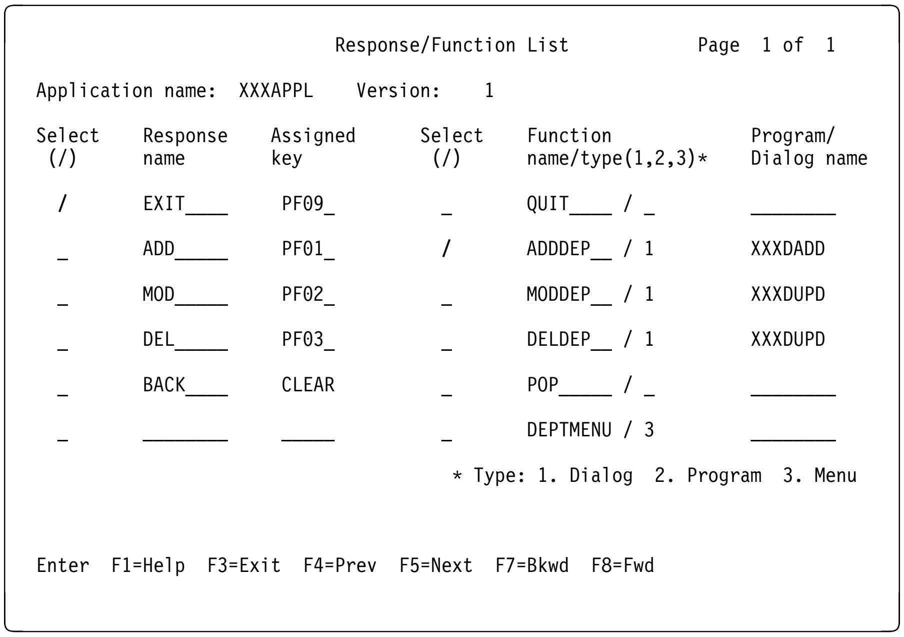
If you made any errors in your application specification, ADSA displays information about another application, and/or displays an error message. In either case, make sure that you typed the correct application name, dictionary, and node, as necessary. You can type over any errors, and then press [Enter] again.

After you successfully retrieve the Department application, you can modify the EXIT response.

10.3.2 Step 2: Select responses and functions

In this step, you will select the response and the function that you want to modify (EXIT response and ADDDEP function).

Response/Function List screen: Choose **2** at the **Screen** prompt on the **Main Menu** screen and press [Enter]. This will bring you to the **Response/Function List** screen. On the Response/Function List screen, select the response and function you want to change.



```
                            Response/Function List          Page  1 of  1

Application name:  XXXAPPL    Version:    1

Select    Response    Assigned      Select    Function            Program/
 (/)      name        key            (/)      name/type(1,2,3)*   Dialog name

  /       EXIT____     PF09_          _       QUIT____ / _        ________

  _       ADD_____     PF01_          /       ADDDEP__ / 1        XXXDADD

  _       MOD_____     PF02_          _       MODDEP__ / 1        XXXDUPD

  _       DEL_____     PF03_          _       DELDEP__ / 1        XXXDUPD

  _       BACK____     CLEAR          _       POP_____ / _        ________

  _       ________     _____          _       DEPTMENU / 3        ________

                                           * Type: 1. Dialog  2. Program  3. Menu


Enter  F1=Help  F3=Exit  F4=Prev  F5=Next  F7=Bkwd  F8=Fwd
```

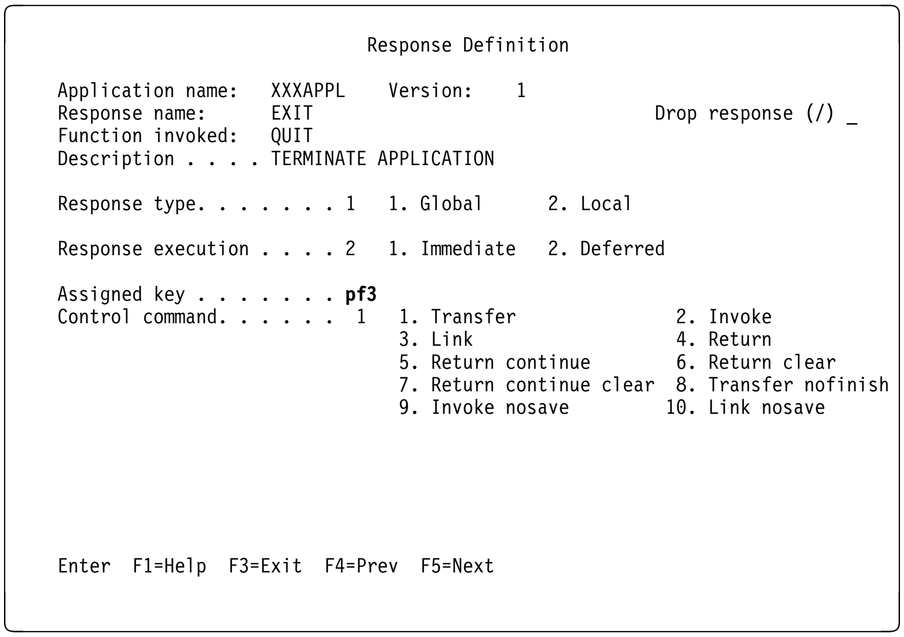
Pressing [PF5] will display the Response Definition screen.

10.3.3 Step 3: Modify the EXIT response

Modifying the EXIT response definition: In this step, you will change the control key for the EXIT response from [PF9] to [PF3]. You use either the **Response/Function List** screen or the **Response Definition** screen to modify an application response assigned key. Use the Response Definition screen to modify the EXIT response in the sample application.

Response Definition screen: Press [PF5] to display the **Response Definition** screen for the EXIT response.

Type the name of the new control key over the previous control key:



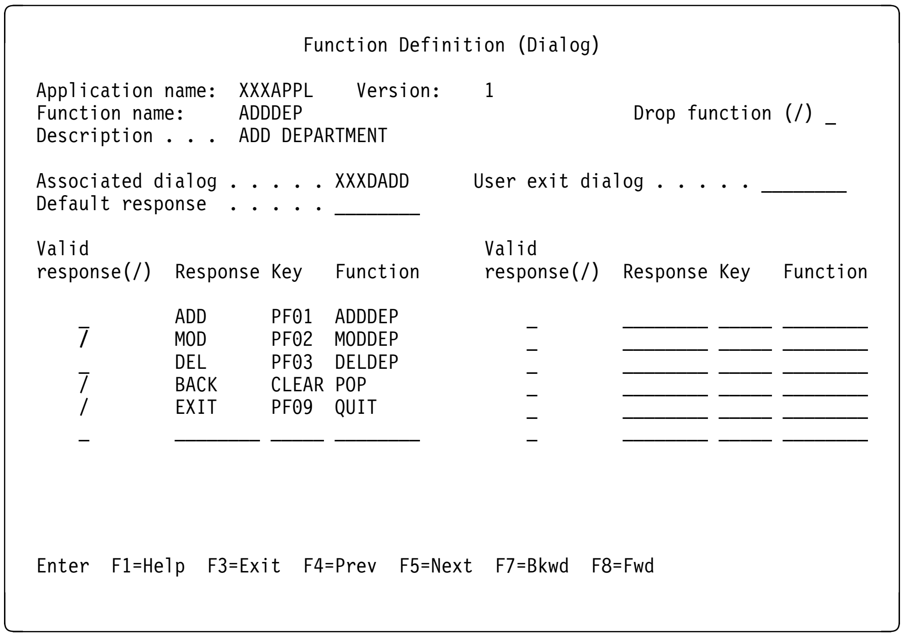
```
                             Response Definition

Application name:   XXXAPPL    Version:    1
Response name:      EXIT                                Drop response (/) _
Function invoked:   QUIT
Description . . . . TERMINATE APPLICATION

Response type. . . . . . . 1   1. Global      2. Local

Response execution . . . . 2   1. Immediate   2. Deferred

Assigned key . . . . . . . pf3
Control command. . . . . .  1   1. Transfer              2. Invoke
                                3. Link                  4. Return
                                5. Return continue       6. Return clear
                                7. Return continue clear  8. Transfer nofinish
                                9. Invoke nosave        10. Link nosave





Enter  F1=Help  F3=Exit  F4=Prev  F5=Next
```

After you modify the EXIT response and press [Enter], ADSA redisplays the Response Definition screen with a confirming message.

As soon as you successfully modify the EXIT response, you can modify the ADDDEP function. Press [PF5] to access the Function Definition screen for the ADDDEP function.

10.3.4 Step 4: Modify the ADDDEP function

Make the MOD response valid from ADDDEP by selecting the response on the Function Definition screen.


```
                            Function Definition (Dialog)

Application name:  XXXAPPL    Version:    1
Function name:     ADDDEP                            Drop function (/) _
Description . . .  ADD DEPARTMENT

Associated dialog . . . . . XXXDADD      User exit dialog . . . . . ________
Default response  . . . . . ________

Valid                                     Valid
response(/)  Response Key   Function      response(/)  Response Key   Function

    _        ADD      PF01  ADDDEP            _        ________ _____ ________
    /        MOD      PF02  MODDEP            _        ________ _____ ________
    _        DEL      PF03  DELDEP            _        ________ _____ ________
    /        BACK     CLEAR POP               _        ________ _____ ________
    /        EXIT     PF09  QUIT              _        ________ _____ ________
    _        ________ _____ ________          _        ________ _____ ________




Enter  F1=Help  F3=Exit  F4=Prev  F5=Next  F7=Bkwd  F8=Fwd
```

Pressing [PF5] brings you back to the Response/Function List since no other responses or functions has been selected. Press [PF3] to return to the Main Menu.

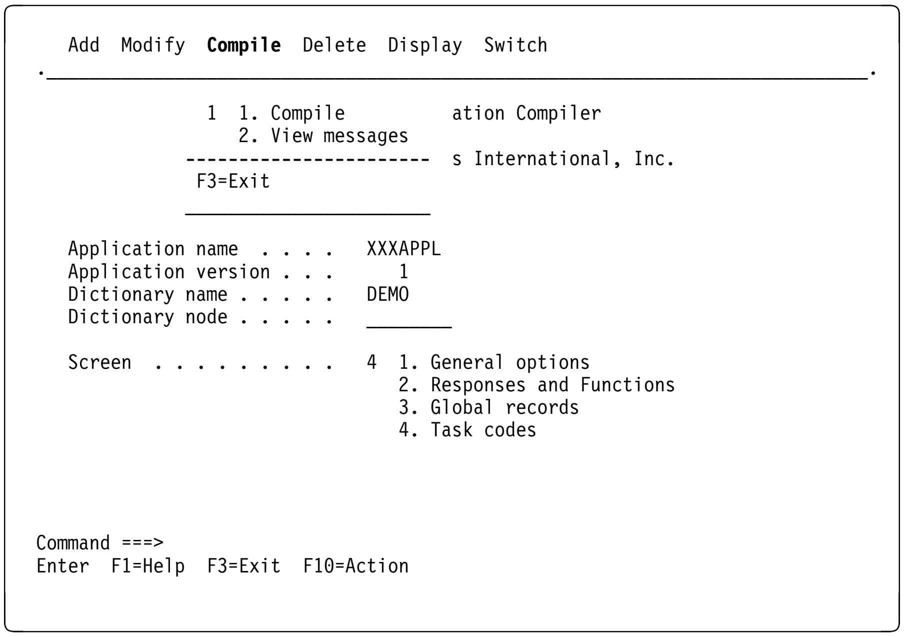
Now you can recompile the application as described below.

10.3.5 Step 5: Recompile the application

After you modify an application, you must recompile the application to create an updated load module for the application. You recompile the Department application by selecting the **Compile** activity.

Compiling the application: To compile the application, position the cursor on the **Compile** item on the action bar and press [Enter]. You can position the cursor on **Compile** by:

- Tabbing to **Compile** and pressing [Enter]
- Pressing [PF10] to move to the action bar and then tabbing to **Compile** and pressing [Enter]
- Typing **compile** on the command line and pressing [Enter]



```
   Add  Modify  Compile  Delete  Display  Switch
.______________________________________________________________________________.

               1  1. Compile         ation Compiler
                  2. View messages
             -----------------------  s International, Inc.
              F3=Exit
             _______________________

   Application name  . . . .   XXXAPPL
   Application version . . .      1
   Dictionary name . . . . .   DEMO
   Dictionary node . . . . .   ________

   Screen  . . . . . . . . .   4  1. General options
                                  2. Responses and Functions
                                  3. Global records
                                  4. Task codes



Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

Once you have displayed the **Compile** action item, press [Enter] to compile the application.

After you press [Enter] to compile the application, ADSA displays either a confirming message to indicate the you successfully compiled the application, or an error message when the application cannot be compiled because of an error.

In the case of compilation errors, you can display diagnostic information by selecting the **View messages** activity from the action bar. Based on this information, you can correct the application and then try to compile the application again.

10.4 Exit from ADSA

You can also return directly to DC/UCF by pressing [PF3]. Alternatively, the **Switch** activity on the action bar of the Main Menu screen allows you to exit from ADSA and transfer to another development tool.

In this sample session, you'll exit to DC/UCF so you can execute your application structure. Press [PF3] to exit.

Note: If you leave ADSA without successfully compiling the current application definition, ADSA saves the suspended definition in a queue record associated with your user ID. In an actual production environment, other users will not be able to access the application definition. To enable them to access the definition, specify the **Release** option from the **Modify** activity on the action bar on the Main Menu.

After you exit from ADSA, you can execute your application as described below.

10.5 Execute the application

In the previous steps, you made the following changes to the structure of the Department application:

- You assigned [PF3] to the EXIT response.
- You made the MOD response valid from the ADDDEP function.

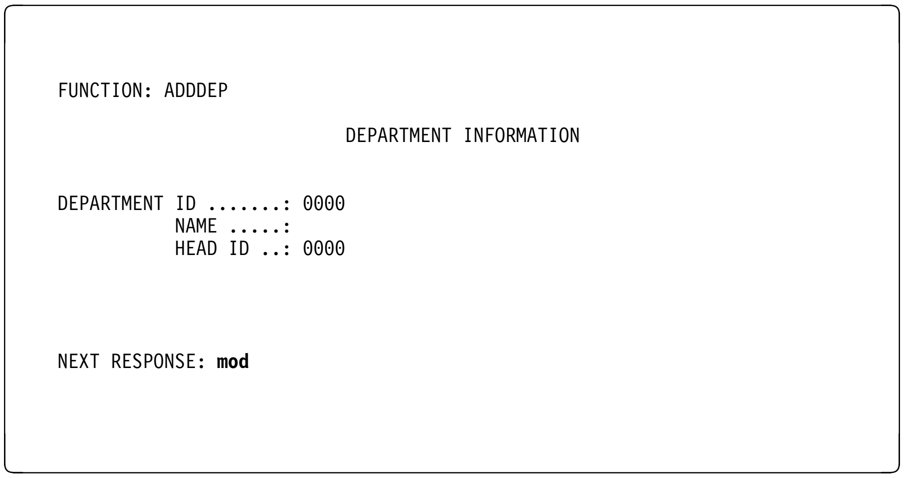
You can now execute the Department application and see how your changes affect the way that end users of the Department application move from one function to another.

You invoke an application from DC/UCF by entering the task code (*XXX*DEPT) for the application. For example, from CA-IDMS/DC, you invoke the Department application as shown:

```
ENTER NEXT TASK CODE:
xxxdept

                                                    [Enter]
```

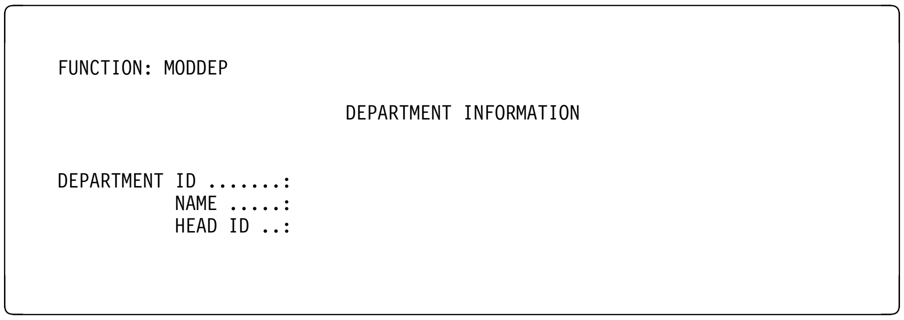
For more information on invoking the Department application, see 9.4, “Instructions for executing the application” on page 9-17.

After you invoke the application, display the ADDDEP function. To test how you transfer from ADDDEP to function MODDEP, you select the MOD response:



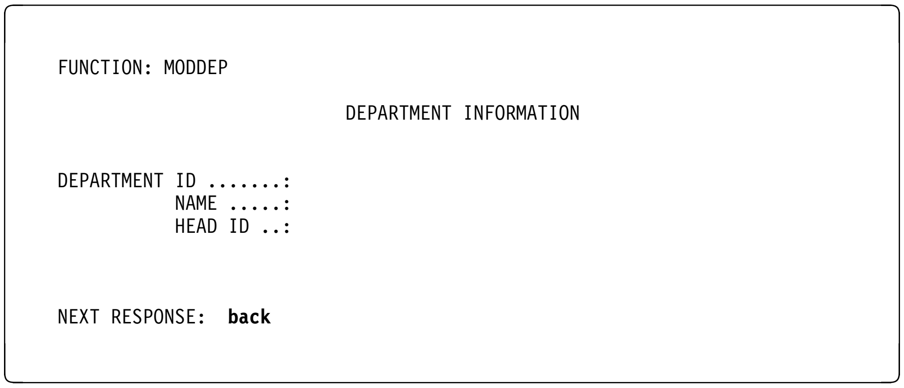
```
FUNCTION: ADDDEP
                           DEPARTMENT INFORMATION


DEPARTMENT ID .......: 0000
           NAME .....:
           HEAD ID ..: 0000



NEXT RESPONSE: mod
```

The MODDEP function is displayed.



```
FUNCTION: MODDEP
                           DEPARTMENT INFORMATION


DEPARTMENT ID .......:
           NAME .....:
           HEAD ID ..:
```

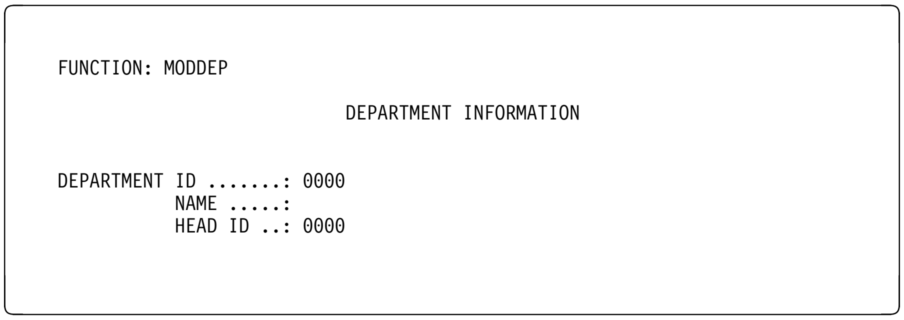
Testing the BACK response: According to your application design, both the BACK and EXIT responses are valid from MODDEP. Try requesting BACK:



```
FUNCTION: MODDEP
                           DEPARTMENT INFORMATION


DEPARTMENT ID .......:
           NAME .....:
           HEAD ID ..:



NEXT RESPONSE:  back
```

Notice that the BACK function takes you to DEPTMENU, rather than to ADDDEP. This is because BACK invokes the POP system function, which returns control to the most recently executed menu in the application.

Pressing [PF3] to invoke the EXIT response: When you are ready to leave the application, you can test out the EXIT response. You can test the EXIT response from any function in the Department application. To test EXIT, try pressing [PF3] to select the EXIT response:



```
FUNCTION: MODDEP
                           DEPARTMENT INFORMATION


DEPARTMENT ID .......: 0000
           NAME .....:
           HEAD ID ..: 0000
```

```
PF3 is now associated --►   [PF3]
with the EXIT response
throughout the application.
```

When you press [PF3], the EXIT response is invoked. The associated system function, QUIT, exits you from the Department application. To test out other features of the Department application, you can invoke the application again, as described earlier.

10.6 Summary

You can use ADSA to modify the application structure at any time. You can use ADSA to add, modify, and delete functions, responses, and task codes, as necessary.

In this chapter, you used ADSA as follows:

1. **You modified the EXIT response** to make [PF3] invoke the EXIT response.
2. **You modified the ADDDEP function** to make the MOD response available directly from ADDDEP.

You can modify any part of the Department application structure at this stage. For example:

- If administrators decide that a particular function is no longer necessary due to changes in regulations, you can use ADSA to quickly delete the function and the response that invokes it.
- If end users decide that a summary or list screen would be useful at some point in the application, you can use ADSA to define a new function to display the summary screen. You can then add a response to invoke the new function, and make other responses valid from the new function.

Additionally, you can modify other application components at any time in an application's life cycle. As an application developer, you can make changes to maps, for example, as soon as end user suggestions are approved. In the next chapter, you will use MAPC to modify map *XXX*MAP.

Chapter 11. Modifying a Map Using MAPC

11.1 Introduction

In the previous chapter, you modified the structure of the Department application based on preferences at the site. Map layouts can also be modified to satisfy end-user and site requirements. As a developer, you can modify a map's layout as soon as the modifications are suggested and approved.

This chapter provides instructions for using MAPC to modify the layout of *XXX*MAP, and includes:

- An overview of modifying maps
- Steps for modifying sample map *XXX*MAP
- Steps for associating the updated map with dialogs that use the map
- Steps for executing the application
- A summary of what you've accomplished in this chapter

11.2 Overview

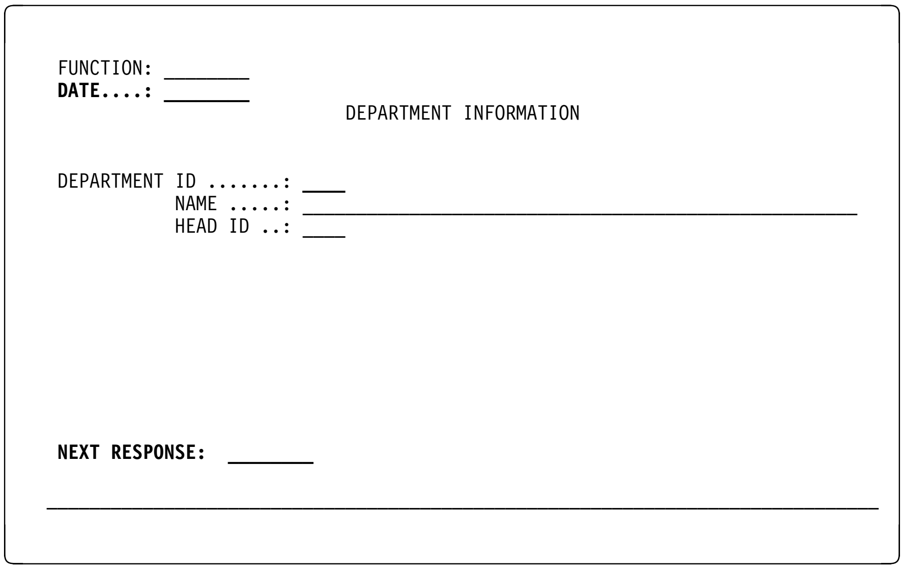
Maps can be modified easily during development or at any other time in an application's life cycle. Modifications can be suggested by development staff and end users. For example, end users who execute the prototype Department application can request that the department ID variable field be displayed in bright intensity to make it easier to locate on the screen.

You can modify a map to make the map conform to screen-display conventions at a given site. For example, it may be necessary to display the current date on each map, or to change the location of particular fields on the map.

Changes to XXXMAP: In this chapter, you will make changes to map *XXX*MAP so that:

- **The current date is displayed on the map** — You will add a variable field to display the current date at runtime, and an adjacent literal field (DATE) to label the displayed information.
- **The department ID number is displayed in bright intensity** — You will modify the variable field that displays department ids to make the data display in bright intensity at runtime.
- **Error messages for fields provide specific information** — You will modify definitions for both the department ID and department head ID variable fields to define specific error messages for the fields.
- **The NEXT RESPONSE literal and variable fields are displayed at a higher row on the screen** — You will modify the NEXT RESPONSE literal and variable fields to move them to row 18 (to make map *XXX*MAP conform to other nonmenu screens at the site).

The following screen shows the layout of the modified map. To modify map *XXX*MAP, you will use the **online mapping facility (MAPC)**, which you used to define the map in Chapter 8, “Defining a Screen Display Using MAPC” on page 8-1. After you modify map *XXX*MAP, you need to update dialogs that use the modified map. To do this, you will use the **CA-ADS dialog compiler (ADSC)**, which you used to define dialogs in Chapter 9, “Defining Dialogs Using ADSC” on page 9-1.



```
FUNCTION: ________
DATE....: ________
                            DEPARTMENT INFORMATION


DEPARTMENT ID .......: ____
           NAME .....: ____________________________________________________
           HEAD ID ..: ____





NEXT RESPONSE:  ________
```

MAPC and ADSC procedures are presented below, followed by a discussion of executing the application.

11.3 Modifying a map using MAPC

In this procedure, you will use MAPC to modify the layout of map *XXX*MAP. You will perform the following steps:

1. Retrieve the map to be modified.
2. Add and select map fields.
3. Edit selected fields.
4. Optionally display the map layout.
5. Recompile the map load module.

These steps are described below. Instructions for associating the modified map definition with dialogs that use the map are presented in 11.9, "Updating modified maps in dialogs using ADSC" on page 11-21, later in this chapter.

11.4 Step 1: Retrieve the map to be modified

In order to modify a map, you must first invoke MAPC.

You can invoke MAPC from CA-IDMS/DC or CA-IDMS/UCF (DC/UCF) by specifying the task code for MAPC (for example, MAPCT) in response to the prompt presented by DC/UCF. For example, when using CA-IDMS/DC, you invoke MAPC as shown:

```
ENTER NEXT TASK CODE:
mapct

                                                   [Enter]
```

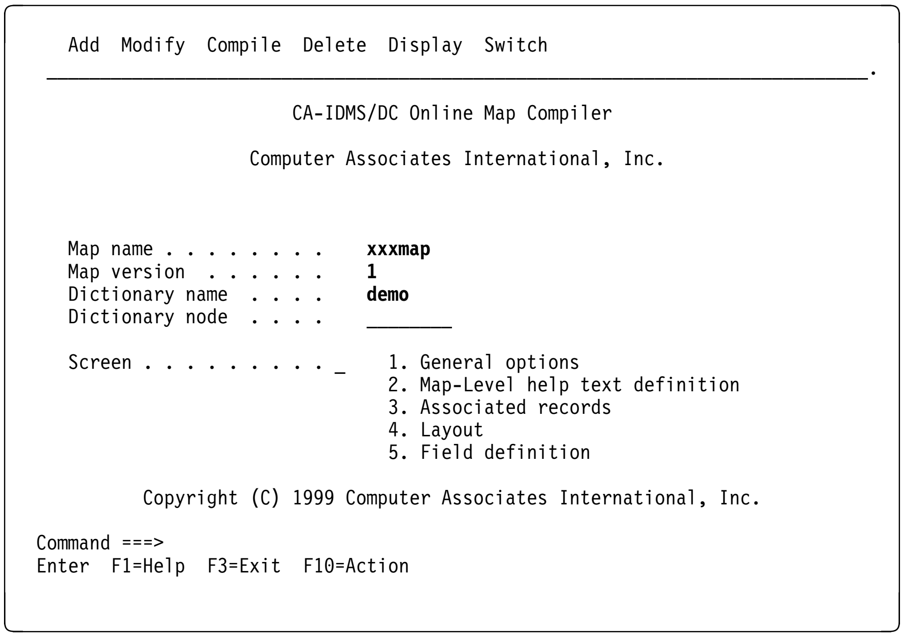
For more information on invoking MAPC, see 8.3.1, “Step 1: Invoke MAPC” on page 8-6.

MAPC begins by displaying the **Main Menu** screen.

Screen prompts: To retrieve a map, you typically enter information after one or more of the following Map Definition screen prompts:

- **Map name** — You must specify the name (*XXX*MAP) that you used when you defined the map in Chapter 8, “Defining a Screen Display Using MAPC” on page 8-1.
- **Dictionary name** — You must specify the same dictionary, if any, as you specified for your map. The correct dictionary name may already be displayed in this field.
- **Dictionary node** — You must specify the same dictionary node, if any, as you specified for your map definition in Chapter 13, “Modifying Process Logic in a Dialog” on page 13-1. The correct dictionary node may already be displayed in this field.

Use the MAPC Main Menu screen to retrieve map *XXX*MAP:



```
   Add  Modify  Compile  Delete  Display  Switch
 _____________________________________________________________________________.

                       CA-IDMS/DC Online Map Compiler

                   Computer Associates International, Inc.



   Map name . . . . . . . .    xxxmap
   Map version  . . . . . .    1
   Dictionary name  . . . .    demo
   Dictionary node  . . . .    ________

   Screen . . . . . . . . . _    1. General options
                                 2. Map-Level help text definition
                                 3. Associated records
                                 4. Layout
                                 5. Field definition

          Copyright (C) 1999 Computer Associates International, Inc.

Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

After you press [Enter], MAPC redisplays the Main Menu screen with a message confirming that the map is available for modification.

Note: If the map has not been explicitly released (using the **Release** option of the **Modify** action on the action bar of the Main Menu), naming the map on the Main Menu screen retrieves that definition for modification. If the map has been released, you use the MAPC Main Menu screen to check out the map definition for modification (using the **Checkout** option of the **Modify** action on the action bar of the Main Menu).

For information on checking out a map, see *CA-IDMS Mapping Facility*.

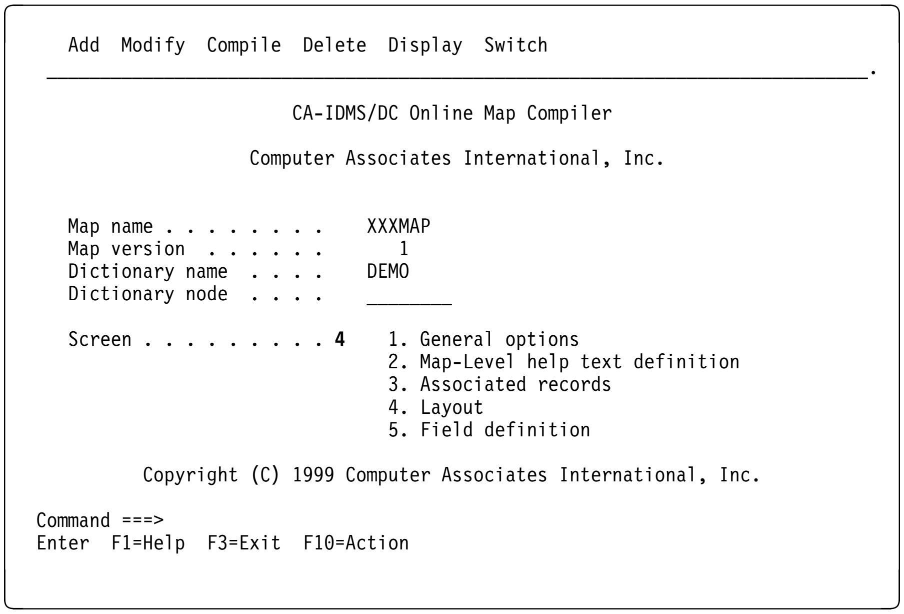
If the map has been released, subsequently checked out to another developer and not released by that developer, you will not be able to check it out.

After you successfully check out *XXX*MAP, you can modify the map layout by adding and selecting map fields.

11.5 Step 2: Add and select map fields

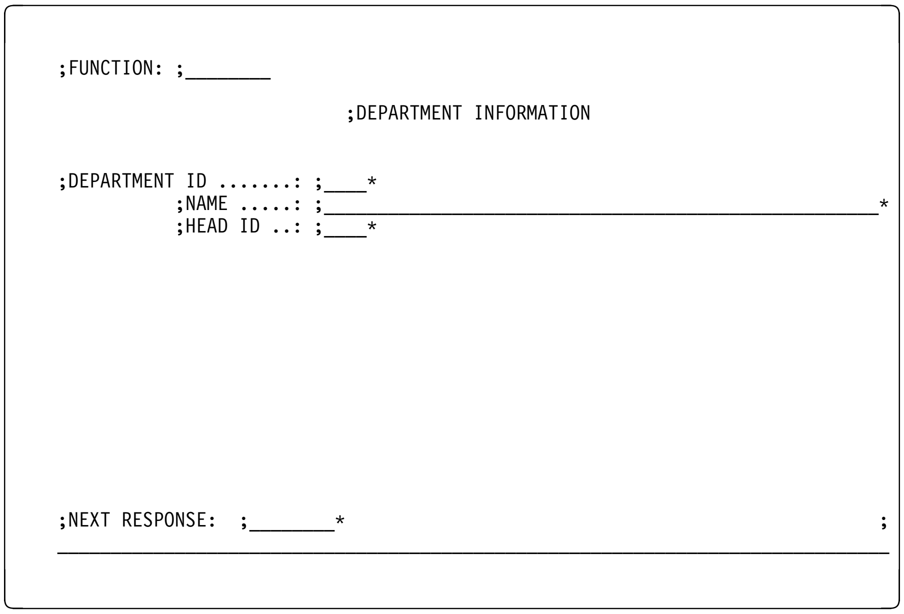
When you defined the layout for *XXX*MAP in 8.3.5, "Step 5: Modify the map layout" on page 8-14, you modified the placement of fields on the map by using the Layout screen. In this chapter, you will use the **Layout** screen to add new fields to the map layout and to select existing fields for modification.

From the Main Menu screen, proceed to the Layout screen entering **4** for the **Screen** prompt and pressing [Enter]:



```
   Add  Modify  Compile  Delete  Display  Switch
 ____________________________________________________________________________.

                      CA-IDMS/DC Online Map Compiler

                  Computer Associates International, Inc.


   Map name . . . . . . . .    XXXMAP
   Map version  . . . . . .       1
   Dictionary name  . . . .    DEMO
   Dictionary node  . . . .    ________

   Screen . . . . . . . . . 4    1. General options
                                 2. Map-Level help text definition
                                 3. Associated records
                                 4. Layout
                                 5. Field definition

          Copyright (C) 1999 Computer Associates International, Inc.

Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

Layout screen: The Layout screen for the map is displayed:



```
;FUNCTION: ;________

                          ;DEPARTMENT INFORMATION


;DEPARTMENT ID .......: ;____*
           ;NAME .....: ;____________________________________________________*
           ;HEAD ID ..: ;____*









;NEXT RESPONSE:  ;________*                                                  ;
______________________________________________________________________________
```

Note: The example above shows the entire map. On the Layout screen, the bottom of the map is hidden by the list of available function keys. Use [PF8] to see the hidden portion of the map.

When you first display the Layout screen for an existing map, each field on the map is preceded by a start-field character, as shown above. While using the Layout screen, you can select a field for editing by:

- Pressing [PF2] while on the field you want to select

  or

- Typing a select-field character (%) in place of the start-field character for the field.

You use the start-field and select-field characters based on the following guidelines:

- The **start-field character** (default is ; or &lbr.) defines the start of a field on the Layout screen.

  Note: On the screens shown in this manual, **;** indicates a start-field character.

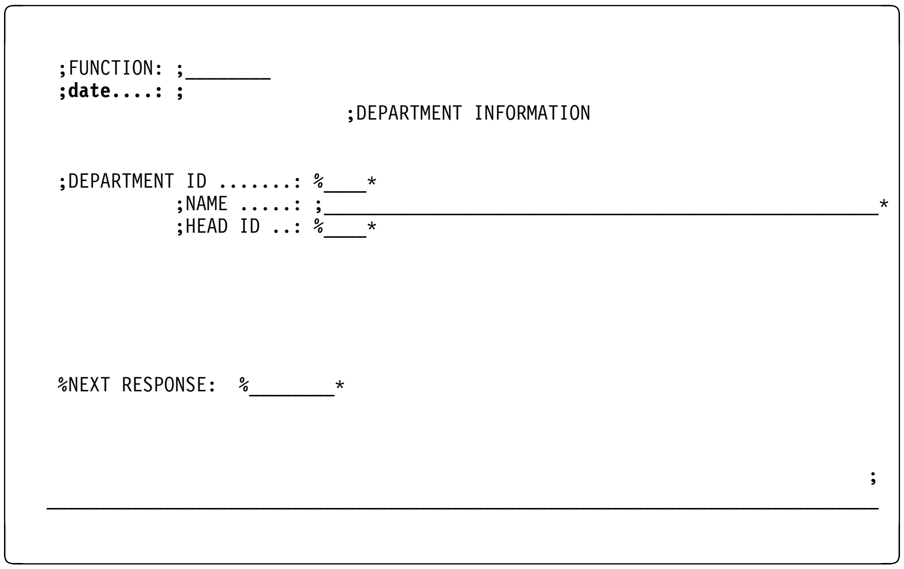
- The **select-field character** (default is %) defines the start of a field and simultaneously selects the field for editing.

You use the Layout screen to modify the layout of map *XXX*MAP and to select fields on the map for further editing:

Modifying the map layout: Make the indicated specifications:

- Begin the new DATE **literal** field with a start-field character (shown here as ;)
- Begin the DATE **variable** field with a select-field character (shown here as ;)

- Type a select-field character (or press [PF2]) over the start-field character for variable fields to select them for editing (DEPARTMENT ID variable field and HEAD ID variable field).
- Select the NEXT RESPONSE **literal and variable** fields for editing by using the select-field character or [PF2].



```
;FUNCTION: ;________
;date....: ;
                           ;DEPARTMENT INFORMATION


;DEPARTMENT ID .......: %____*
           ;NAME .....: ;____________________________________________________*
           ;HEAD ID ..: %____*




%NEXT RESPONSE:  %________*


                                                                              ;
________________________________________________________________________________
```

After you press [Enter], MAPC redisplays the Layout screen so that you can inspect the screen. You can correct any errors on the Layout screen in either of the following ways:

- **To change a few fields**, type over the characters that you want to change and press [Enter] again.
- **To erase the modifications that you just made**, press the CLEAR key.

  In this chapter, pressing CLEAR on the Layout screen does not erase fields that you defined when you originally generated the map in Chapter 8, “Defining a Screen Display Using MAPC” on page 8-1.

11.6 Step 3: Edit the selected fields

To edit new and existing fields in this chapter, you will use the Field and Literal Definition screens to make specifications for fields, such as the display intensity of the runtime field, define error messages for the department ID and department head ID variable field, and so forth.

You selected fields for editing by using the Layout screen earlier in this chapter, in 11.5, “Step 2: Add and select map fields” on page 11-9. The following table summarizes the specifications that you will make when editing each selected field. You will edit each selected field's definition as indicated in this table.

For more information on using prompts to edit field definitions, see 8.3.5, “Step 5: Modify the map layout” on page 8-14.

Location of field on map	Purpose of field	Specifications for field

After DATE literal field (in the upper right corner)	Displays the current date	`Element name: AGR-DATE`[1]
		`Protected`
		
		`Edit Picture  XX/XX/XX`

After DEPARTMENT ID literal field	Displays a department's unique ID number	`Bright display`
		`Error message:`
		`*ENTER A NUMERIC`
		`DEPT ID*`

After HEAD ID literal field	Displays the ID number for the head of the department	`Error message:`
		`*ENTER A NUMERIC`
		`DEPT ID*`

Literal NEXT RESPONSE in lower left corner	Prompts the user to input a response name	`ROW .......... : 18`
After NEXT RESPONSE literal field	Allows a user to input a response name	`ROW .......... : 18`

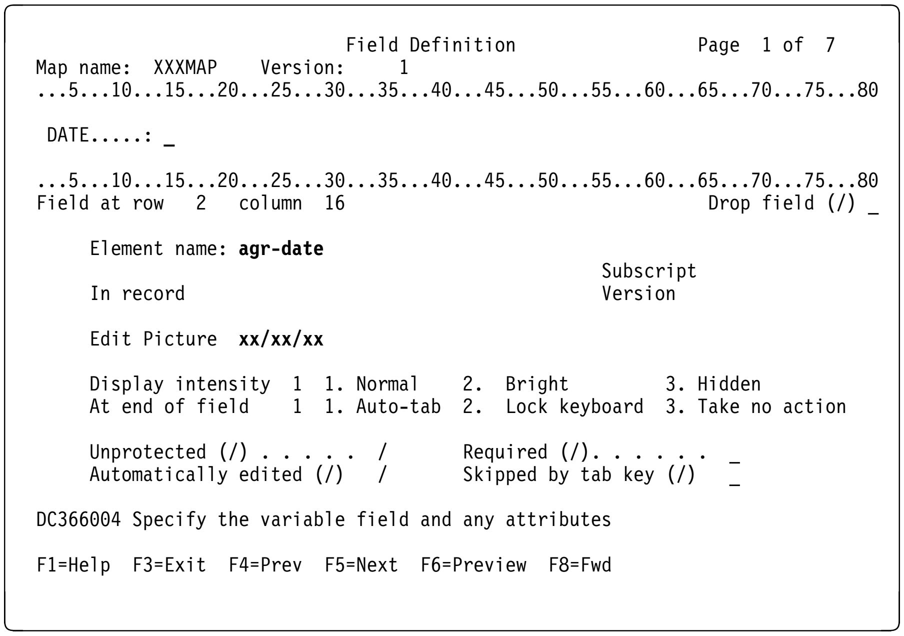
[1] AGR-DATE is an element in ADSO-APPLICATION-GLOBAL-RECORD

Press [PF5] from the Layout screen to begin editing fields. This brings you to the first definition screen, a Field Definition screen for the DATE variable field.

Editing the DATE variable field: On the Field Definition screen, you edit the DATE variable field information as shown:

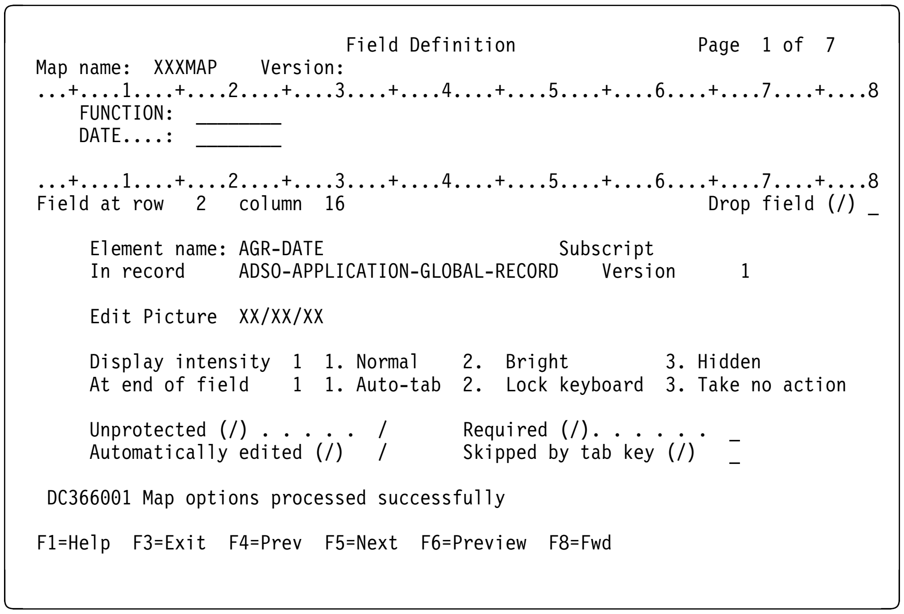
- The field being edited is highlighted on the screen.
- Name the record element (in this example, AGR-DATE) to be associated with the variable field.

- Specify an edit picture of **xx/xx/xx**.



```
                             Field Definition                 Page  1 of  7
Map name:  XXXMAP    Version:     1
...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80

 DATE.....: _

...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80
Field at row   2   column  16                                  Drop field (/) _

     Element name: agr-date
                                                  Subscript
     In record                                    Version

     Edit Picture  xx/xx/xx

     Display intensity  1  1. Normal    2.  Bright        3. Hidden
     At end of field    1  1. Auto-tab  2.  Lock keyboard  3. Take no action

     Unprotected (/) . . . . .  /      Required (/). . . . . .  _
     Automatically edited (/)   /      Skipped by tab key (/)   _

DC366004 Specify the variable field and any attributes

F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F8=Fwd
```

When you press [Enter], the Field Definition screen is redisplayed with a confirming message.



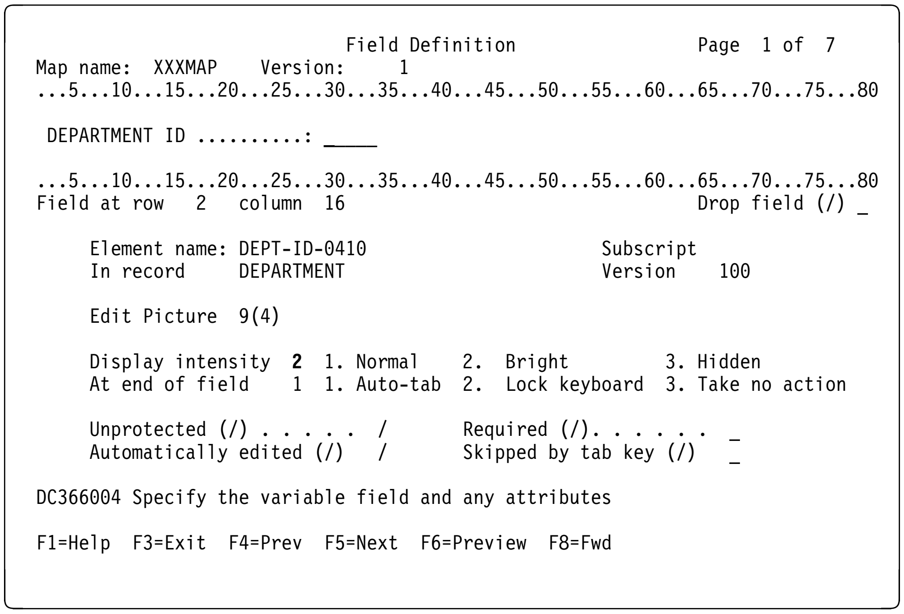
```
                             Field Definition                 Page  1 of  7
Map name:  XXXMAP    Version:
...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
    FUNCTION:  ________
    DATE....:  ________

...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
Field at row   2   column  16                                  Drop field (/) _

     Element name: AGR-DATE                      Subscript
     In record     ADSO-APPLICATION-GLOBAL-RECORD    Version      1

     Edit Picture  XX/XX/XX

     Display intensity  1  1. Normal    2.  Bright        3. Hidden
     At end of field    1  1. Auto-tab  2.  Lock keyboard  3. Take no action

     Unprotected (/) . . . . .  /      Required (/). . . . . .  _
     Automatically edited (/)   /      Skipped by tab key (/)   _

 DC366001 Map options processed successfully

F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F8=Fwd
```

This message informs you that the *new* field definition contains no errors and has been successfully added to the map. Underscores displayed for the field show you the length of the variable field.

Editing the DEPARTMENT ID variable field: The next map field selected for editing is the DEPARTMENT ID variable field. To edit this field, you will:

1. Make data in the field display in bright intensity.
2. Define an error message for the field. To do this, you will use page 3 - **Additional Edit Criteria** - of the Field Definition screen.

Pressing [PF5] brings the Field Definition screen highlighting the variable field for the department ID.



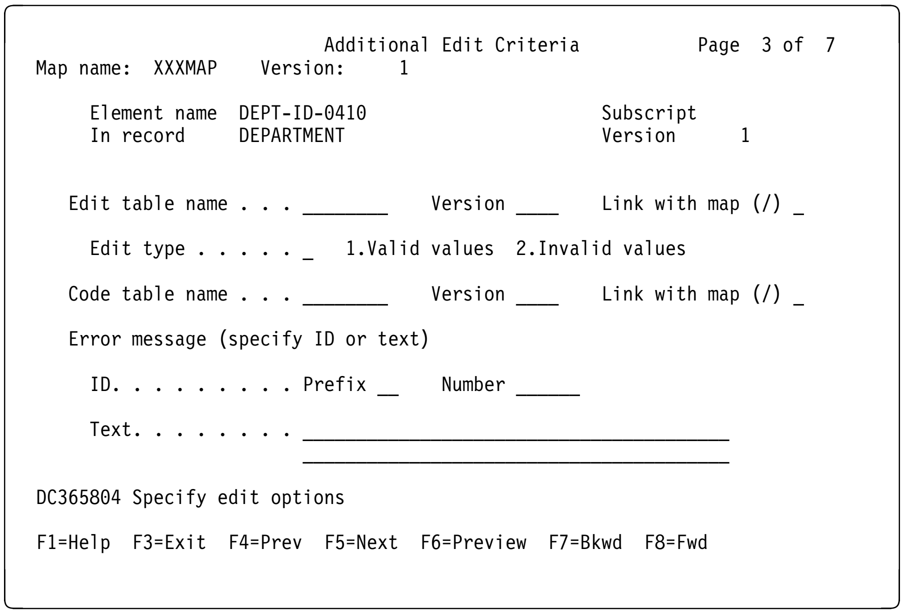
```
                             Field Definition                 Page  1 of  7
Map name:  XXXMAP    Version:     1
...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80

 DEPARTMENT ID ..........: _____

...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80
Field at row   2   column  16                                 Drop field (/) _

     Element name: DEPT-ID-0410                      Subscript
     In record     DEPARTMENT                        Version    100

     Edit Picture  9(4)

     Display intensity  2  1. Normal    2.  Bright        3. Hidden
     At end of field    1  1. Auto-tab  2.  Lock keyboard  3. Take no action

     Unprotected (/) . . . . . .  /      Required (/). . . . . .  _
     Automatically edited (/)    /      Skipped by tab key (/)    _

DC366004 Specify the variable field and any attributes

F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F8=Fwd
```

When you press [Enter] to input your modification, the Field Definition screen displays a confirming message.

This message informs you that an *existing* field definition contains no errors and has been successfully modified in the map.

Defining an error message: To define a message for this field, press [PF8] twice (or change the page number in the upper right corner) to get to page 3 - **Additional Edit Criteria** - of the Field Definition screen.

Sample Additional Edit Criteria screen

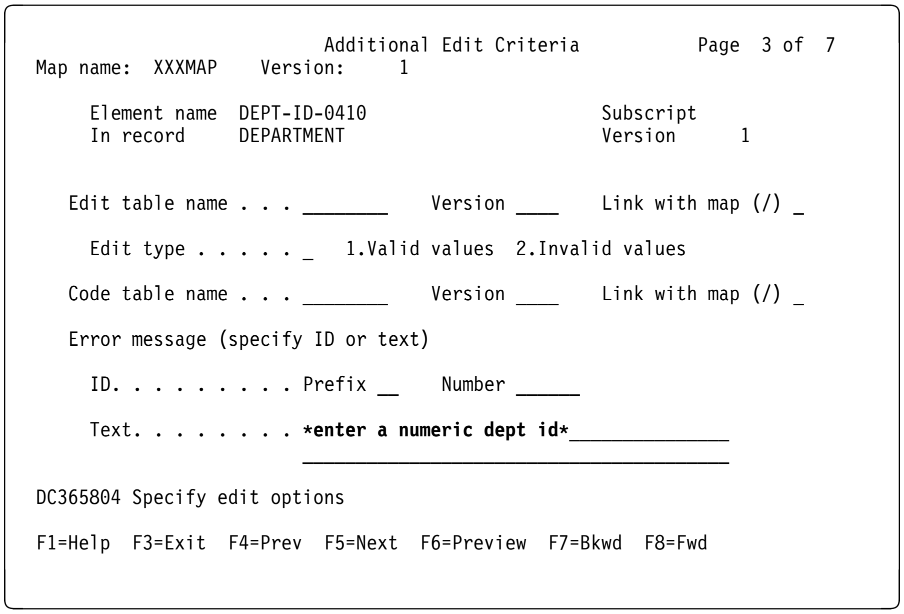


```
                              Additional Edit Criteria          Page  3 of  7
Map name:  XXXMAP    Version:     1

     Element name  DEPT-ID-0410                      Subscript
     In record     DEPARTMENT                        Version      1


   Edit table name . . . ________    Version ____    Link with map (/) _

     Edit type . . . . . _   1.Valid values  2.Invalid values

   Code table name . . . ________    Version ____    Link with map (/) _

   Error message (specify ID or text)

     ID. . . . . . . . . Prefix __    Number ______

     Text. . . . . . . . ________________________________________
                         ________________________________________

DC365804 Specify edit options

F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F7=Bkwd  F8=Fwd
```

Screen prompts

Error message — Enter the text for the message that will display for the field when an error is encountered.

Add the error message associated with the DEPARTMENT ID variable field as shown:

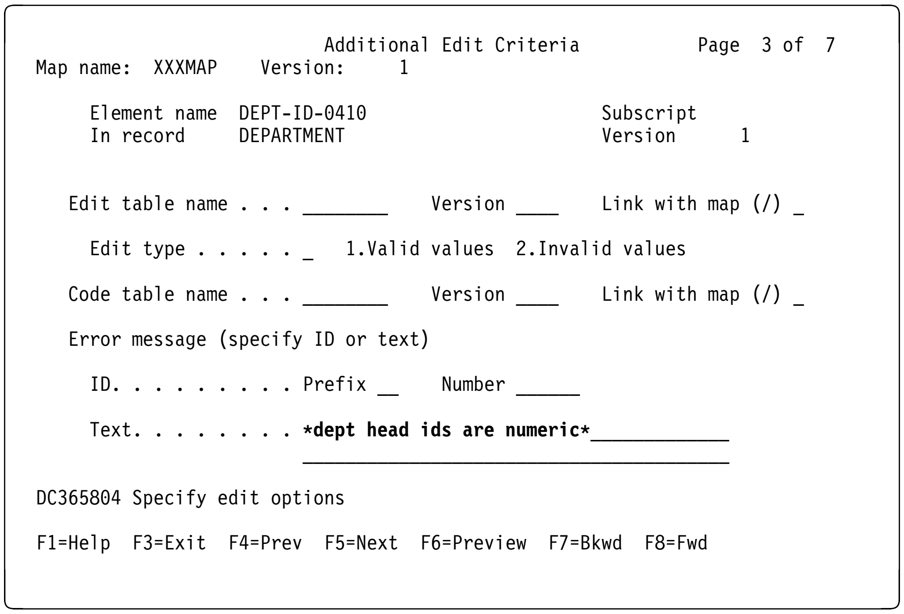


```
                              Additional Edit Criteria          Page  3 of  7
Map name:  XXXMAP    Version:     1

     Element name  DEPT-ID-0410                      Subscript
     In record     DEPARTMENT                        Version      1


   Edit table name . . . ________    Version ____    Link with map (/) _

     Edit type . . . . . _   1.Valid values  2.Invalid values

   Code table name . . . ________    Version ____    Link with map (/) _

   Error message (specify ID or text)

     ID. . . . . . . . . Prefix __    Number ______

     Text. . . . . . . . *enter a numeric dept id*_______________
                         ________________________________________

DC365804 Specify edit options

F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F7=Bkwd  F8=Fwd
```

A confirming message is displayed when your specifications on the Additional Edit Criteria screen contain no errors and have modified the field definition.

Press [PF5] to have the Field Definition screen for the next select field displayed.

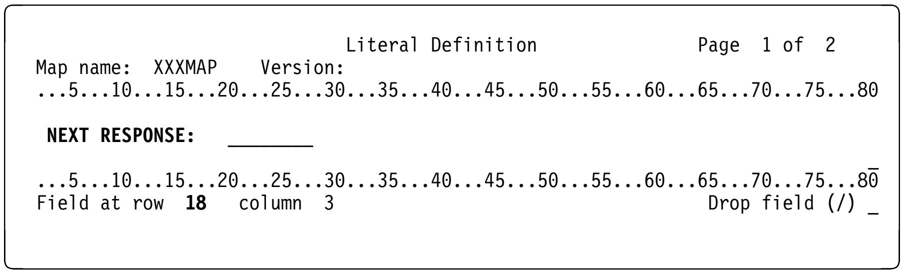
Editing the HEAD ID variable field: The next field selected for editing is the HEAD ID variable field. You need to define an error message for this field by using the **Additional Edit Criteria** screen.



```
                         Additional Edit Criteria          Page  3 of  7
Map name:  XXXMAP    Version:     1

     Element name  DEPT-ID-0410                     Subscript
     In record     DEPARTMENT                       Version      1


   Edit table name . . . ________    Version ____    Link with map (/) _

     Edit type . . . . . _   1.Valid values  2.Invalid values

   Code table name . . . ________    Version ____    Link with map (/) _

   Error message (specify ID or text)

     ID. . . . . . . . . Prefix __    Number ______

     Text. . . . . . . . *dept head ids are numeric*_____________
                         ________________________________________

DC365804 Specify edit options

F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F7=Bkwd  F8=Fwd
```

When you have successfully defined an error message for the DEPT-HEAD-ID variable field, you can edit the next selected field. To display the next selected field, press [PF5]. MAPC displays the Literal Definition screen for the NEXT RESPONSE literal field.

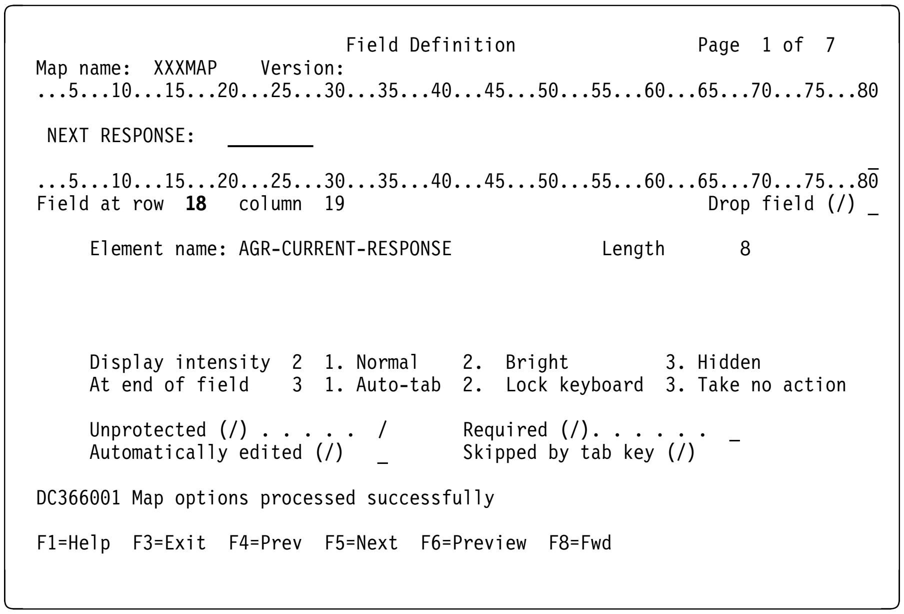
Displaying and moving the NEXT RESPONSE literal field: To move the NEXT RESPONSE literal field, change the row number to reflect the new position. (Remember that you could have used the Layout screen with the alternate PF keys to move both this literal field and its variable field.)



```
                           Literal Definition              Page  1 of  2
Map name:  XXXMAP    Version:
...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80

 NEXT RESPONSE:   ________

...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80
Field at row  18   column  3                                    Drop field (/) _
```

After you press [Enter], the NEXT RESPONSE literal field is displayed in its new location.

Displaying and moving the NEXT RESPONSE variable field: Press [PF5] to see the next field to be defined. The next field is the NEXT RESPONSE variable field. Move this field to row 18.



```
                             Field Definition                Page  1 of  7
Map name:  XXXMAP    Version:
...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80

 NEXT RESPONSE:   ________

...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80
Field at row  18   column  19                                  Drop field (/) _

     Element name: AGR-CURRENT-RESPONSE              Length       8




     Display intensity  2  1. Normal    2.  Bright         3. Hidden
     At end of field    3  1. Auto-tab  2.  Lock keyboard  3. Take no action

     Unprotected (/) . . . . . .  /      Required (/). . . . . .  _
     Automatically edited (/)    _       Skipped by tab key (/)

DC366001 Map options processed successfully

F1=Help  F3=Exit  F4=Prev  F5=Next  F6=Preview  F8=Fwd
```

After you press [Enter], the Field Definition screen for the response variable field is redisplayed with a confirming message.

You have edited all of the fields you selected for editing when you used the Layout screen earlier in this chapter, in 11.5, “Step 2: Add and select map fields” on page 11-9.

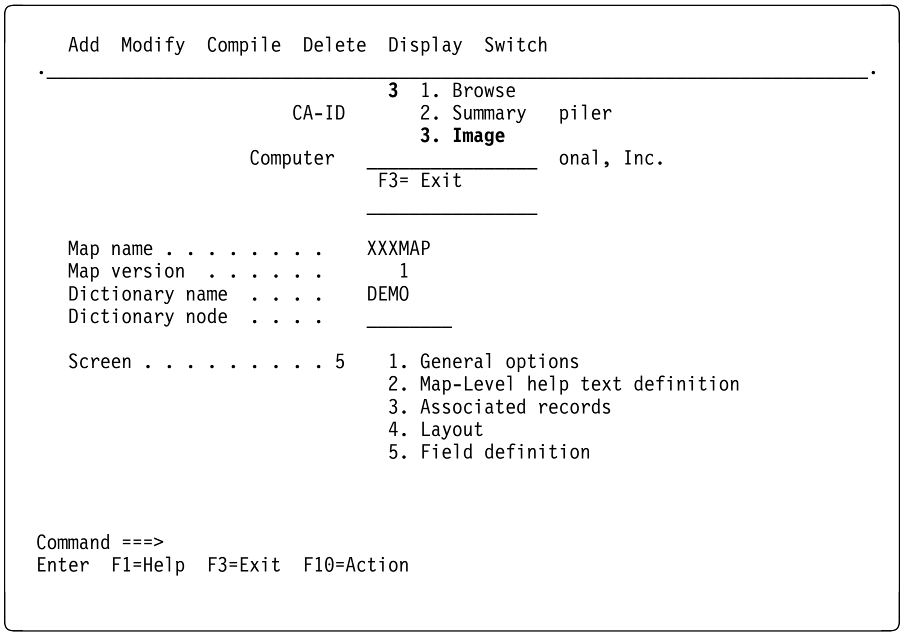
Pressing [PF5] brings you to Main Menu screen.

Before you recompile the map, you can display the current layout for map *XXX*MAP.

11.7 Step 4: Optionally display the map layout

You can use the **Map Image screen** in MAPC to see how the modified map will look to a user. This allows you to see how modifications affect a map layout before you recompile the map load module.

You can display the Map Image screen by selecting **Image** option from the **Display** activity on the action bar:



```
   Add  Modify  Compile  Delete  Display  Switch
._____________________________________________________________________________.
                                 3  1. Browse
                       CA-ID        2. Summary   piler
                                    3. Image
                   Computer     ________________  onal, Inc.
                                 F3= Exit
                                ________________

   Map name . . . . . . . .     XXXMAP
   Map version  . . . . . .        1
   Dictionary name  . . . .     DEMO
   Dictionary node  . . . .     ________

   Screen . . . . . . . . . 5     1. General options
                                  2. Map-Level help text definition
                                  3. Associated records
                                  4. Layout
                                  5. Field definition



Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

If you note any errors while displaying the Map Image screen, you can quickly correct the map while still using MAPC. You can then redisplay the Map Image screen to check your corrections.

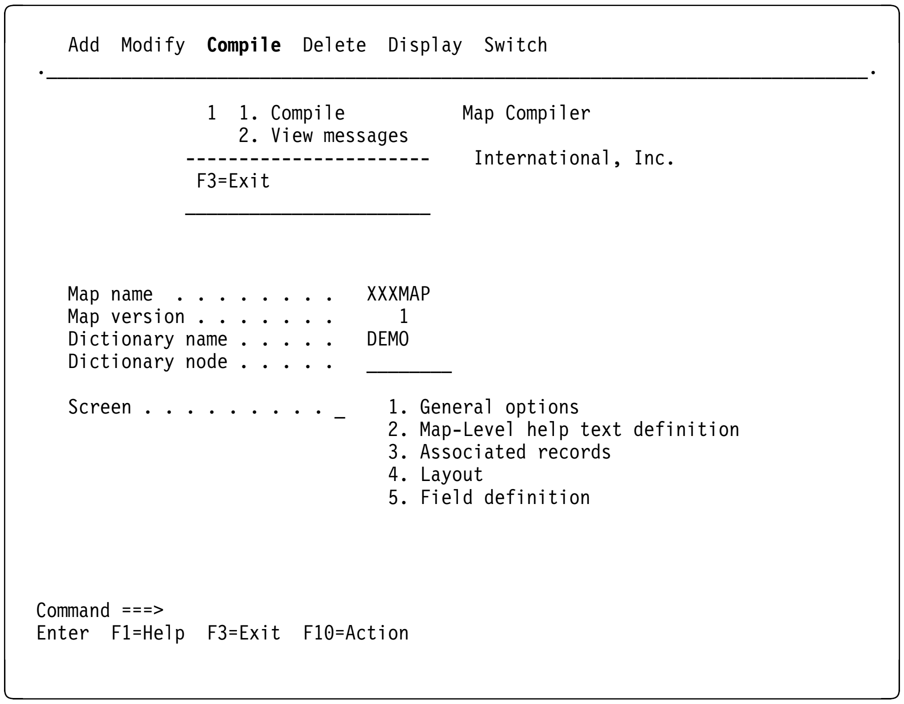
When you are satisfied with the modified map layout as displayed on the Map Image screen, you can recompile the map.

Press [PF3] from the Map Image screen to return to the Main Menu.

11.8 Step 5: Recompile the map

By using MAPC screens, you modified map *XXX*MAP by adding new map fields and changing existing fields. You now need to create an updated load module for the map. You compile the *XXX*MAP map as shown:

Compiling the map



```
   Add  Modify  Compile  Delete  Display  Switch
.____________________________________________________________________________.

             1  1. Compile          Map Compiler
                2. View messages
           -----------------------    International, Inc.
            F3=Exit
           _______________________



   Map name  . . . . . . . .   XXXMAP
   Map version . . . . . . .      1
   Dictionary name . . . . .   DEMO
   Dictionary node . . . . .   ________

   Screen . . . . . . . . . _    1. General options
                                 2. Map-Level help text definition
                                 3. Associated records
                                 4. Layout
                                 5. Field definition



Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

To compile the application, position the cursor on the **Compile** item on the action bar and press [Enter]. You can position the cursor on **Compile** by:

- Tabbing to **Compile** and pressing [Enter]
- Pressing [PF10] to move to the action bar and then tabbing to **Compile** and pressing [Enter]
- Typing **compile** on the command line and pressing [Enter]

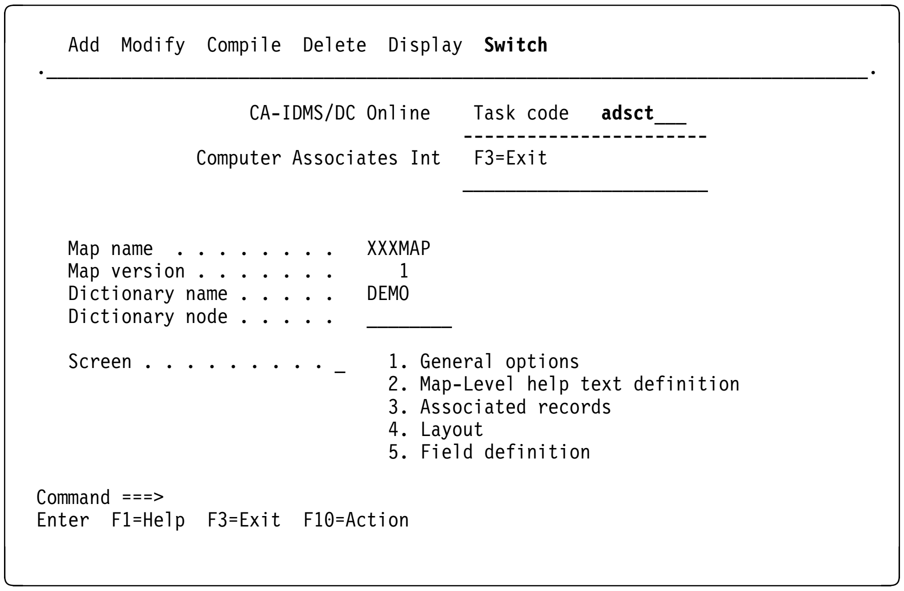
Once you have displayed the **Compile** action item, press [Enter] to compile the map.

When you press [Enter] to recompile the map, MAPC displays messages to indicate whether the map load module recompiled successfully.

- **If the map recompiled successfully**, MAPC a confirming message.
- **If the map could not be recompiled**, MAPC displays an error message. In this case, read the message to determine the problem. After correcting the errors, try again to recompile the map.

Exit from MAPC: After you successfully compile your map, you can exit from MAPC by using the **Switch** activity on the Main Menu screen. In this example, you use **Switch** to transfer directly to ADSC to associate modified map *XXX*MAP with dialogs that use the map:

Selecting the Switch activity



```
   Add  Modify  Compile  Delete  Display  Switch
.______________________________________________________________________________.

                    CA-IDMS/DC Online    Task code   adsct___
                                        -----------------------
               Computer Associates Int   F3=Exit
                                        _______________________


   Map name  . . . . . . . .   XXXMAP
   Map version . . . . . . .      1
   Dictionary name . . . . .   DEMO
   Dictionary node . . . . .   ________

   Screen . . . . . . . . . _    1. General options
                                 2. Map-Level help text definition
                                 3. Associated records
                                 4. Layout
                                 5. Field definition

Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

In this example, ADSCT is the sample task code for ADSC.

Note: You cannot use the switch action unless you entered MAPC using the transfer control facility task code (MAPCT).

11.9 Updating modified maps in dialogs using ADSC

After you modify map *XXX*MAP, you use ADSC to associate the modified map with dialogs *XXX*DADD and *XXX*DUPD. To do this, you will perform the following steps:

1. Retrieve dialog *XXX*DADD.
2. Recompile the dialog load module.
3. Retrieve and recompile dialog *XXX*DUPD.

11.10 Step 1: Retrieve dialog XXXDADD

In order to retrieve a dialog load module, you use ADSC.

If you *did not* transfer directly to ADSC earlier in this chapter when you exited from MAPC, you need to invoke ADSC by using the task code (for example, ADSCT) for ADSC.

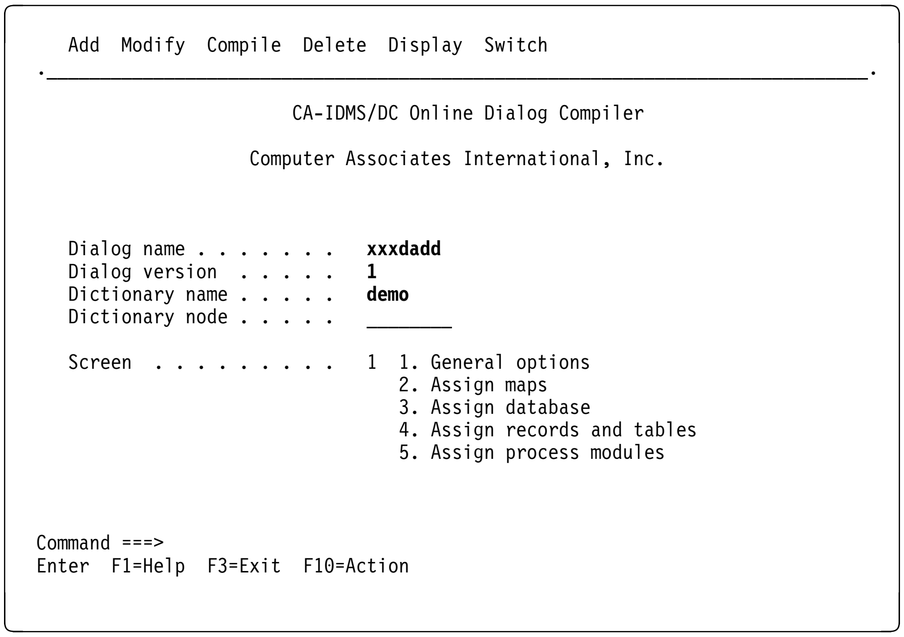
Note: Using the task code ADSCT means that you are invoking ADSC under TCF (the transfer control facility). Once you are in the ADSC tool under TCF, you can switch to another application development tool without returning to DC/UCF.

For more information on invoking ADSC, see 9.3.1, “Step 1: Invoke ADSC” on page 9-8.

ADSC begins by displaying the **Main Menu** screen. You use the Main Menu screen to retrieve a dialog definition for update.

Screen prompts: You typically enter information after one or more of the following Main Menu screen prompts:

- **Dialog name** — You must specify the name (*XXX*DADD) that you used when you defined the dialog in 9.3.2, “Step 2: Define dialog XXXDADD” on page 9-9.
- **Dictionary name** — You must specify the same dictionary, if any, as you specified for your dialog in Chapter 9, “Defining Dialogs Using ADSC” on page 9-1. The correct dictionary name may already be displayed in this field.
- **Dictionary node** — You must specify the same dictionary node, if any, as you specified for your dialog. The correct dictionary node may already be displayed in this field.



```
   Add  Modify  Compile  Delete  Display  Switch
.______________________________________________________________________________.
                        CA-IDMS/DC Online Dialog Compiler

                    Computer Associates International, Inc.



   Dialog name . . . . . . .   xxxdadd
   Dialog version  . . . . .   1
   Dictionary name . . . . .   demo
   Dictionary node . . . . .   ________

   Screen  . . . . . . . . .   1  1. General options
                                  2. Assign maps
                                  3. Assign database
                                  4. Assign records and tables
                                  5. Assign process modules



Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

Press [Enter] to retrieve the dialog.

Note: If the dialog has not been explicitly released (using the **Release** option of the **Modify** action on the action bar of the Main Menu), naming the dialog on the Main Menu screen retrieves that definition for modification. If the dialog has been released, you use the ADSC Main Menu screen to check out the dialog definition for modification (using the **Checkout** option of the **Modify** action on the action bar of the Main Menu).

For information on checking out a dialog, see *CA-ADS Reference*.

If the dialog has been released, subsequently checked out to another developer and not released by that developer, you will not be able to check it out.

After you press [Enter], ADSC redisplays the Main Menu screen with either a confirming message or an error message.

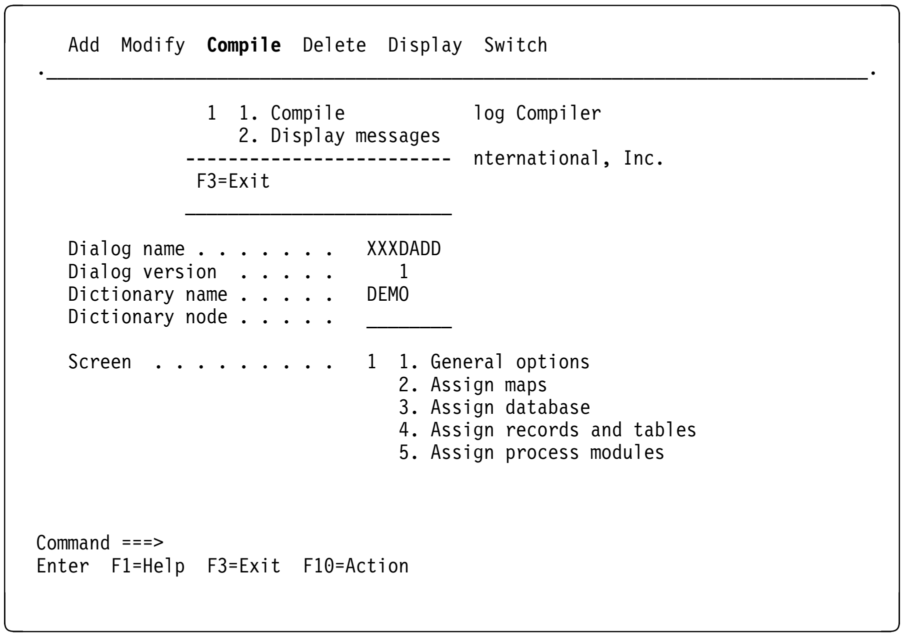
Make sure that you typed the correct dialog name, dictionary, and node, as necessary. You can type over any errors and then press [Enter] again.

After you successfully retrieve *XXX*DADD, you can recompile the dialog load module as described below.

11.11 Step 2: Recompile dialog XXXDADD

To include modified map *XXX*MAP in dialog *XXX*DADD, you need only recompile the dialog:

Compiling the dialog: You compile the *XXX*DADD dialog as shown:



```
   Add  Modify  Compile  Delete  Display  Switch
._______________________________________________________________.

               1  1. Compile           log Compiler
                  2. Display messages
             -------------------------  nternational, Inc.
              F3=Exit
             _________________________

   Dialog name . . . . . . .   XXXDADD
   Dialog version  . . . . .      1
   Dictionary name . . . . .   DEMO
   Dictionary node . . . . .   ________

   Screen  . . . . . . . . .   1  1. General options
                                  2. Assign maps
                                  3. Assign database
                                  4. Assign records and tables
                                  5. Assign process modules



Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

To compile the application, position the cursor on the **Compile** item on the action bar and press [Enter]. You can position the cursor on **Compile** by:

- Tabbing to **Compile** and pressing [Enter]
- Pressing [PF10] to move to the action bar and then tabbing to **Compile** and pressing [Enter]
- Typing **compile** on the command line and pressing [Enter]

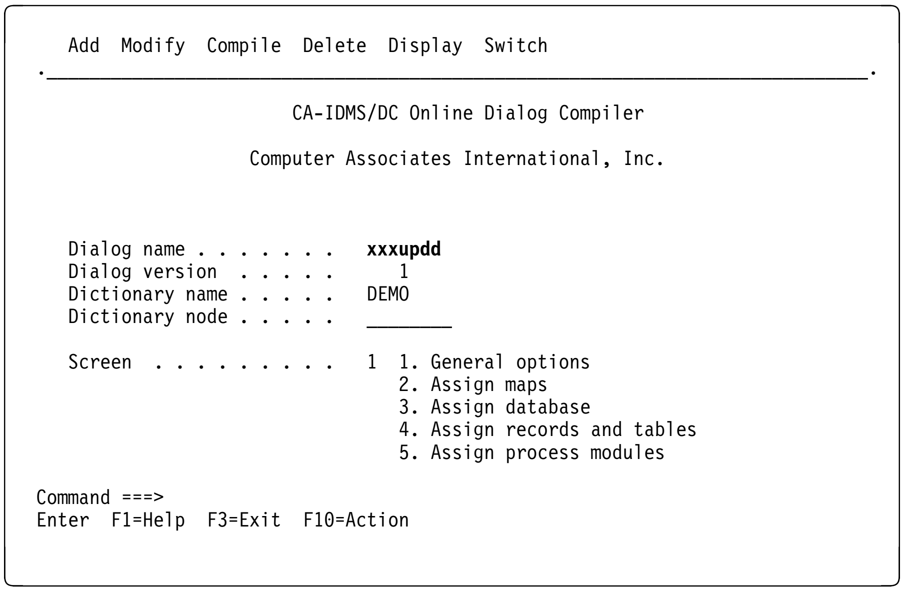
Once you have displayed the **Compile** action item, press [Enter] to compile the dialog.

When you press [Enter], ADSC attempts to recompile the dialog load module using the new map. ADSC displays a message to indicate whether the dialog has been successfully recompiled or whether errors are present. In the case of errors, read the message to determine the problem. Use ADSC to correct any errors, and then recompile the dialog as described above.

After you successfully recompile dialog *XXX*DADD, you can retrieve and recompile dialog *XXX*DUPD as described below.

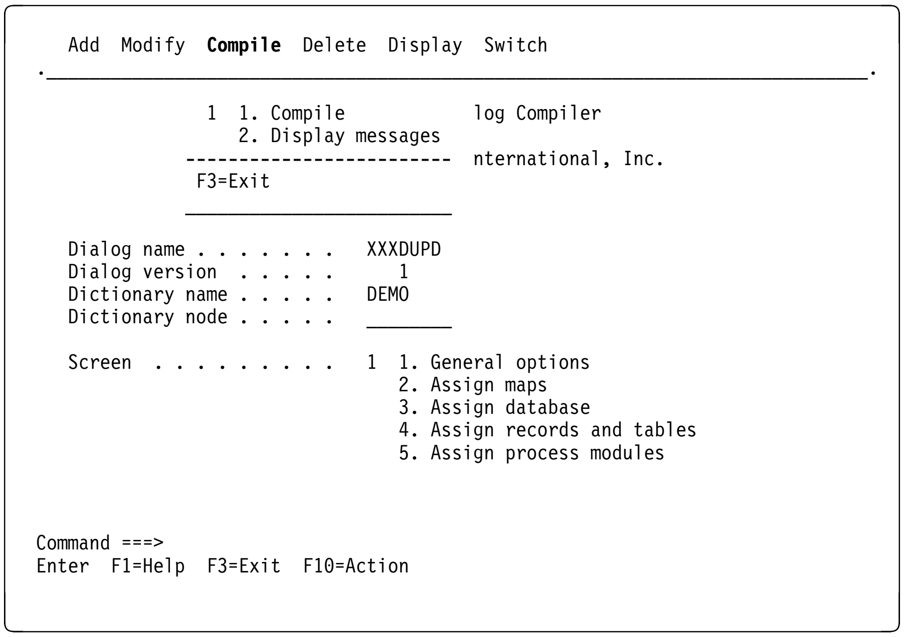
11.12 Step 3: Retrieve and recompile dialog XXXDUPD

You display dialog *XXX*DUPD by overtyping the **Dialog name** on the **Main Menu** screen and retrieving it.


```
   Add  Modify  Compile  Delete  Display  Switch
.______________________________________________________________________________.
                      CA-IDMS/DC Online Dialog Compiler

                  Computer Associates International, Inc.



   Dialog name . . . . . . .   xxxupdd
   Dialog version  . . . . .      1
   Dictionary name . . . . .   DEMO
   Dictionary node . . . . .   ________

   Screen  . . . . . . . . .   1  1. General options
                                  2. Assign maps
                                  3. Assign database
                                  4. Assign records and tables
                                  5. Assign process modules

Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

When you have successfully retrieving the dialog, you can immediately recompile the dialog load module.

Compiling the dialog: You compile the *XXX*DADD dialog as shown:


```
   Add  Modify  Compile  Delete  Display  Switch
.______________________________________________________________________________.
              1  1. Compile           log Compiler
                 2. Display messages
            -------------------------  nternational, Inc.
             F3=Exit
            _________________________

   Dialog name . . . . . . .   XXXDUPD
   Dialog version  . . . . .      1
   Dictionary name . . . . .   DEMO
   Dictionary node . . . . .   ________

   Screen  . . . . . . . . .   1  1. General options
                                  2. Assign maps
                                  3. Assign database
                                  4. Assign records and tables
                                  5. Assign process modules



Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

After you successfully recompile dialog *XXX*DUPD, you can exit from ADSC and return to DC/UCF to execute the application. You press [PF3] from the ADSC Main Menu to exit from ADSC.

11.13 Executing the application

At some sites, modified maps are not automatically loaded in the program pool when an old copy of the map already is in the pool.

Note: The new copy of a map load module is loaded automatically in the program pool at your site if the system generation program specifies that NEW COPY IS YES for OLM The default is NEW COPY IS NO.

This strategy allows users who are executing the old copy of the map to complete their work. When no users are executing the map, developers can make the modified map available for execution. As an application developer, you do this by updating the map in dialogs that use the map and then dynamically loading the modified map in the program pool.

You included modified map *XXX*MAP in dialogs *XXX*DADD and *XXX*DUPD above, in 11.9, "Updating modified maps in dialogs using ADSC" on page 11-21. If modified maps are not automatically loaded in the program pool at your site, you now need to dynamically load modified map *XXX*MAP in the program pool.

Dynamically loading a modified map in the program pool is discussed below, followed by steps for invoking and executing the application.

11.14 Optionally loading the modified map

You can manually load modified maps in the program pool by issuing a DCMT VARY PROGRAM NEW COPY command. If you are not sure whether modified maps are automatically loaded at your site, go ahead and issue this DCMT command for your modified map.

Note: Do not confuse these commands:

- **DCMT VARY PROGRAM** loads a modified map.
- **DCMT VARY** *DYNAMIC* **PROGRAM** dynamically redefines characteristics of programs and maps and can interfere with your ability to execute the map.

You enter the DCMT VARY PROGRAM command in response to the DC/UCF prompt displayed at your site. The following example shows how to enter this command while using CA-IDMS/DC:

```
ENTER NEXT TASK CODE:
dcmt vary program demo..xxxmap new copy.

                                                            [Enter]
```

If the DCMT command is **successful**, DC/UCF displays a confirming message.

If a different message is displayed, verify that you specified the correct map in the DCMT command. You can type the DCMT command again using the correct map name.

Note:

> If there are multiple components with the same name, you will be presented with a list of the components and asked to identify the one to be varied.

If you *did* type the correct map name, you can proceed to execute the Department application. In this case, the program pool doesn't contain a copy of the old map, so a new copy will be loaded automatically the next time a dialog displays the map.

11.15 Invoking and executing the application

To invoke the Department application from DC/UCF, you enter the task code (*XXX*DEPT) for the application. For example, when using CA-IDMS/DC, you invoke the Department application as shown:

```
ENTER NEXT TASK CODE:
xxxdept
                                                    [Enter]
```

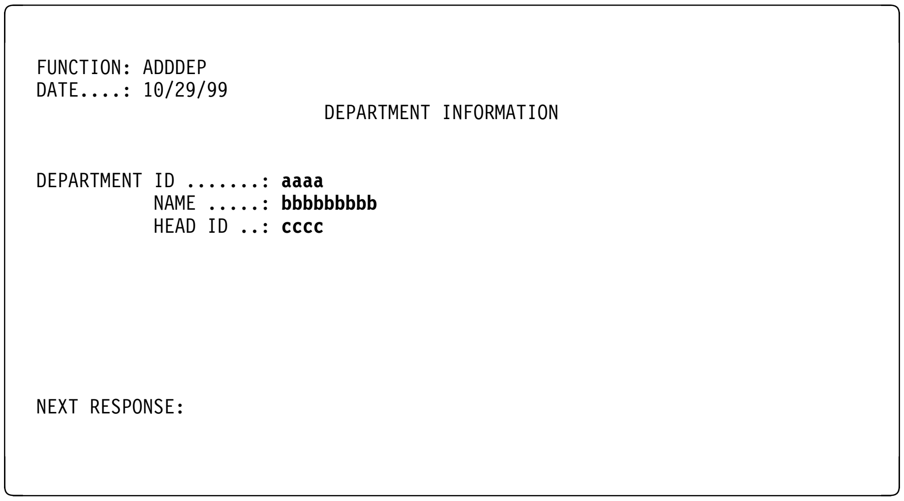
For more information on invoking the Department application, see 9.4, "Instructions for executing the application" on page 9-17.

To test modifications you made to map *XXX*MAP, you need to execute either dialog *XXX*DADD or *XXX*DUPD. According to your application design, each of the following responses selects a function that executes either dialog *XXX*DADD or *XXX*DUPD:

- **ADD** selects dialog function ADDDEP. In the final application, ADDDEP will allow end users to add new department information. ADDDEP executes dialog *XXX*DADD.
- **MOD** selects dialog function MODDEP, which will allow end users to modify existing department information. MODDEP executes dialog *XXX*DUPD.
- **DEL** selects dialog function DELDEP, which will allow end users to delete existing department information. DELDEP executes dialog *XXX*DUPD.

Testing error messages: To quickly test out error messages that you defined for map *XXX*MAP, you can invoke function ADDDEP. The modified map, *XXX*MAP, is displayed for the function ADDDEP.

Try entering invalid department and department head ID numbers as shown:

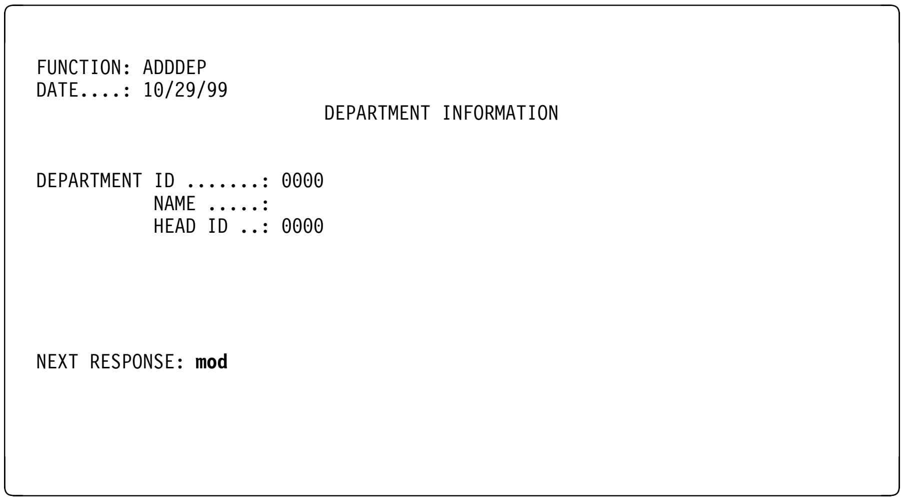


```
FUNCTION: ADDDEP
DATE....: 10/29/99
                           DEPARTMENT INFORMATION


DEPARTMENT ID .......: aaaa
           NAME .....: bbbbbbbbb
           HEAD ID ..: cccc








NEXT RESPONSE:
```

When you press [Enter], your error messages are displayed in the map's message field. For example:

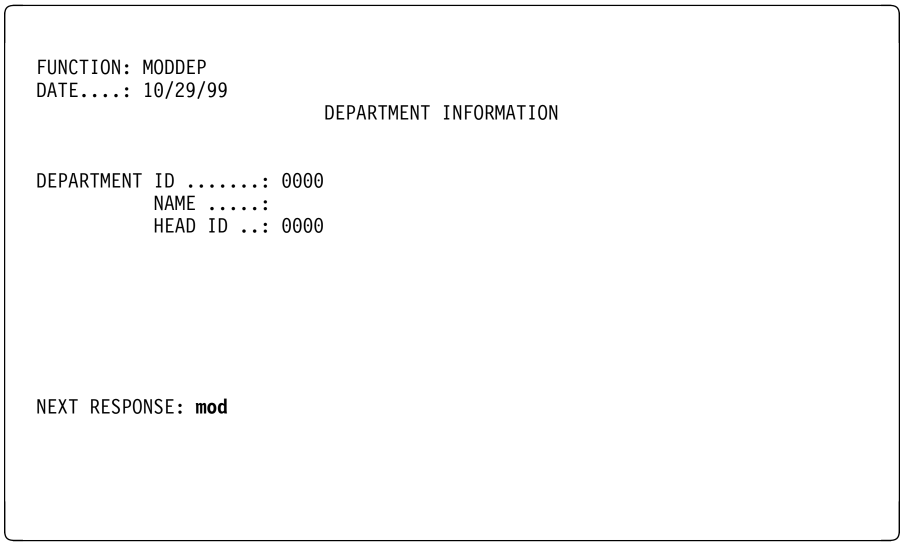
```
* ENTER A NUMERIC DEPT ID * * DEPT HEAD IDS ARE NUMERIC *
```

Verify the modified map: You can verify that functions MODDEP and DELDEP also display modified map *XXX*MAP by displaying these functions. Try transferring from ADDDEP to MODDEP:



```
FUNCTION: ADDDEP
DATE....: 10/29/99
                           DEPARTMENT INFORMATION


DEPARTMENT ID .......: 0000
           NAME .....:
           HEAD ID ..: 0000





NEXT RESPONSE: mod
```

The MODDEP function is displayed. MODDEP also displays modified map, *XXX*MAP.



```
FUNCTION: MODDEP
DATE....: 10/29/99
                           DEPARTMENT INFORMATION


DEPARTMENT ID .......: 0000
           NAME .....:
           HEAD ID ..: 0000






NEXT RESPONSE: mod
```

You also can transfer to function DELDEP if you want.

Exit from the application: To exit from the Department application, you can use the EXIT response anywhere in the application.

11.16 Summary

You can use MAPC to modify maps during development or at any other time in an application's life cycle. For example, you can modify a map:

- To display information that becomes available as the application is expanded. For example, the name of each department head can be displayed on map *XXX*MAP when the Department application includes employee data.
- To collect additional information. For example, new government or tax regulations can require the collection of new data.
- To display fields in conformity with end-user requests and site conventions. For example, you modified map *XXX*MAP in this chapter for these reasons.

In this chapter, you used **MAPC** to modify the layout of map *XXX*MAP as described below:

1. **You added two new map fields on the Layout screen**. You added the DATE literal field and an adjacent variable field.
2. **You selected fields for editing on the Layout screen**. You used the select-field character (default is %) to select the new variable field and four existing map fields for editing.
3. **You edited the five selected fields on the Field Definition and Literal Definition screens**. You associated the new variable field with AGR-DATE to display the current calendar date in the runtime field.

   You modified the variable field for department ids so that ids display in bright intensity at runtime. You modified the department ID field and the department head id field to define specific error messages for these fields. You then moved the RESPONSE literal field and the adjacent variable field to a different location on the map.

When you recompiled map *XXX*MAP, MAPC informed you of a critical change on the map. When you make a critical change to a map, you must recompile dialogs that use the map.

In this chapter, you used **ADSC** to recompile dialogs *XXX*DADD and *XXX*DUPD, which both use map *XXX*MAP. After you exited from ADSC, you optionally issued a DCMT VARY PROGRAM NEW COPY command for map *XXX*MAP to load the new map load module in the program pool for execution.

No process statements: You then executed the prototype application. Notice that you do not need to write any process statements to develop a working prototype for an application. This is true regardless of the application's size. An application's user interface can be defined, tested, and tailored for your users before you have any process logic to modify.

End users and application development staff can execute the prototype and implement changes until the prototype is approved. You can then use CA-ADS development tools to enhance the application so that it can be used to store, display, modify, and delete data in the application database.

As the application developer, you will enhance the Department application by writing process code and creating work records for dialogs, as described in Part III of this manual.

Chapter 12. Adding Process Logic to a Dialog

12.1 Introduction

You defined a prototype Department application in Part II of this manual. You executed this prototype to test out the flow of control between application functions at runtime. Executing the prototype enabled you to see how screens are displayed to users.

Because you have not yet written any process commands to access the database, the prototype Department application does not retrieve or store data in the database. For example, sample department data that you type on the screen while viewing the ADDDEP function does not get stored in the database when you press [Enter].

In CA-ADS, components of a prototype application can be developed directly into the production application. In Part III of this manual, you will complete the Department application so that it is fully functional. To do this, you will add modules of process commands to dialogs *XXX*DADD and *XXX*DUPD, which you defined in Part II. The process modules that you define for these dialogs will allow users to store, display, modify, and delete department records in the database.

In this chapter, you will define the two process modules required for dialog *XXX*DADD. This chapter includes:

- An overview of defining process modules for dialogs
- Steps for defining process modules
- Steps for adding process modules to dialogs
- Steps for executing the application
- A summary of what you've accomplished in this chapter

12.2 Overview

Because CA-ADS is a fourth-generation application development system, major portions of an CA-ADS application can be defined without writing any code. For example, you defined the entire Department application prototype in Part II of this manual without writing any code.

To enable dialogs to perform runtime processing, you define modules of process commands for the dialogs. For example, you can define process modules to retrieve and display database information, to display messages, to receive input from users, and to evaluate and store valid data.

Process language: You write process modules by using the CA-ADS **process language**. This language incorporates all the processing capabilities found in a traditional programming language. For example, you can evaluate strings, perform arithmetic functions, and perform conditional tests and loops. Additionally, the CA-ADS process language benefits from complete integration with the CA-ADS environment.

Categories of process commands: The following table lists categories of process commands. For detailed information on process commands, see the *CA-ADS Reference*.

Category of command	Capabilities
Arithmetic and assignment	Perform calculations and move data
Conditional	Perform testing and looping
Control	Specify the next application component executed, govern data passed to that component, and display maps
Database	Perform database retrieval and update functions and specify recovery options
Map modification	Request temporary or permanent changes to a map at runtime
Pageable map	Create, display, and retrieve sets of fields (detail occurrences) for a pageable map
Queue and scratch management	Define and access temporary disk storage
Subroutine control	Define and call subroutines
Utility	Retrieve runtime system status information, request memory dumps, initialize record buffers, direct output to a printer, and display diagnostic information

Process modules: Process modules can be executed before and after the dialog's map is displayed to the user.

Premap process: The process module is called a **premap process** when executed before the map. A dialog can have a maximum of one premap process.

A premap process typically includes commands that prepare the map for display. For example, commands in a premap process can retrieve stored values from the database and then display the values along with a message on the dialog's map.

Response process: The process module is called a **response process** when executed after the map. A dialog can have any number of response processes. The response process executed at runtime is determined by actions that the user takes when inputting data on the map.

A response process typically includes commands that accept end-user input for evaluation and storage.

Declaration module: A dialog can have a maximum of one declaration module. The modules is not executed, but contains declaration statements for SQL that are used during dialog compilation.

>> For information on declaration statements, see *CA-IDMS SQL Programming*.

Accessing the database: Some process modules access the database; for example, to store a new department record.

Process modules accessing a non-SQL defined database can use SQL DML statements or non-SQL DML statements to access that data. Process modules accessing an SQL-defined database can use SQL DML statements.

A dialog using non-SQL DML statements to access a non-SQL defined database must know which portion of the database to access at runtime. You supply this information by adding a predefined subset of the database (that is, a **subschema**) with the dialog. The subschema that you name for dialog *XXX*DADD, for example, identifies the portion of the sample database that contains the DEPARTMENT record. Subschemas usually are defined by database administrators (DBAs) at a site.

A dialog that uses SQL statements to access an SQL-defined (or non-SQL defined) database will access **tables** that have been defined through SQL statements. To execute the SQL statements, an **access module** must be created based on the SQL statements in one or more programs. The access module is created **after** dialog compilation; it does not have to be predefined.

>> For further information on programming using SQL DML statements, see *CA-IDMS SQL Programming* and *CA-IDMS SQL Reference*. For further information on programming using non-SQL DML statements, see *CA-ADS Reference* and *CA-IDMS Navigational DML Programming*.

Work records: You also can have **work records** associated with a map or associated directly with a dialog. A work record does not participate in a subschema. Data defined by work records is not stored in the database. You will not add a work record to dialog *XXX*DADD.

The following figure shows the components of a fully developed dialog:

- A premap process is executed before the dialog's map is displayed to the user.
- A response process is executed when the user inputs data on the map.
- A subschema specifies the subset of the application database (schema) that is available to the dialog.
- Work records define data that is used at runtime but not stored in the database.

Integrated Data Dictionary: As an application developer, you define premap and response processes by using the **Integrated Data Dictionary (IDD) menu facility**.

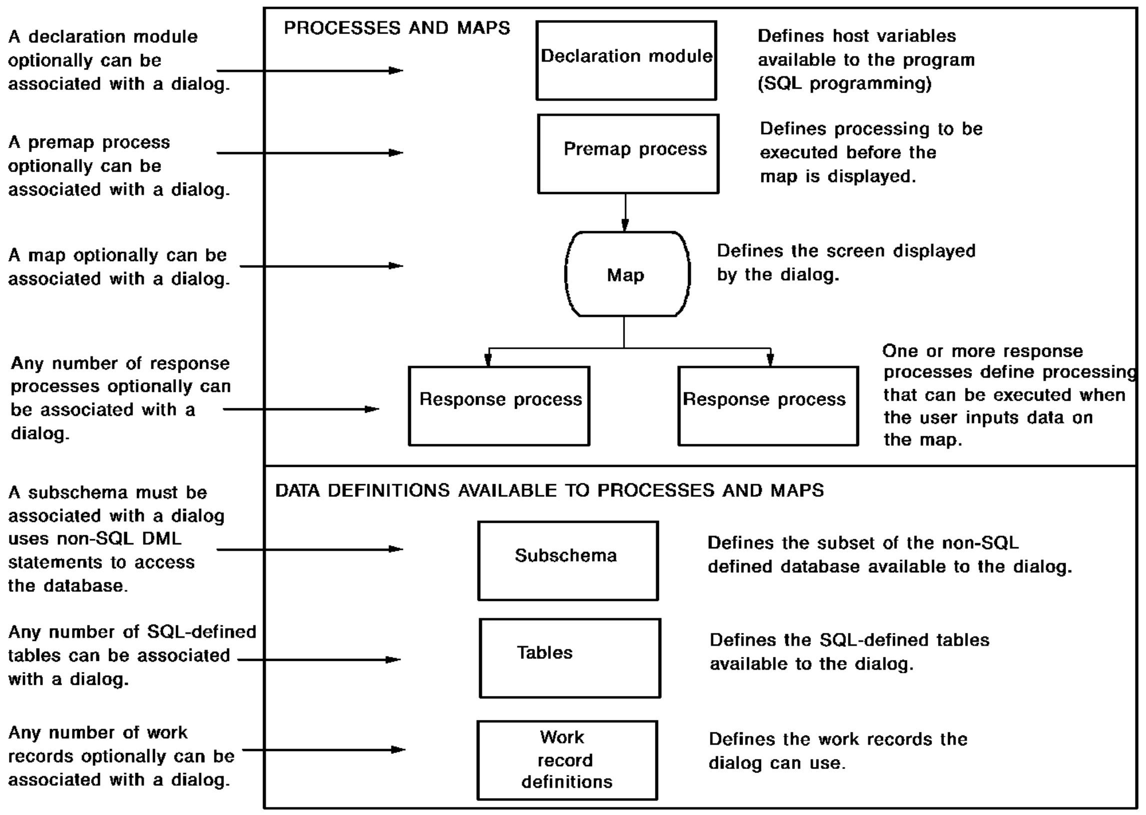
You will use IDD in this chapter to define one premap process (*XXX*DADD-PREMAP) and one response process (*XXX*DADD-RESPONSE) for dialog *XXX*DADD:

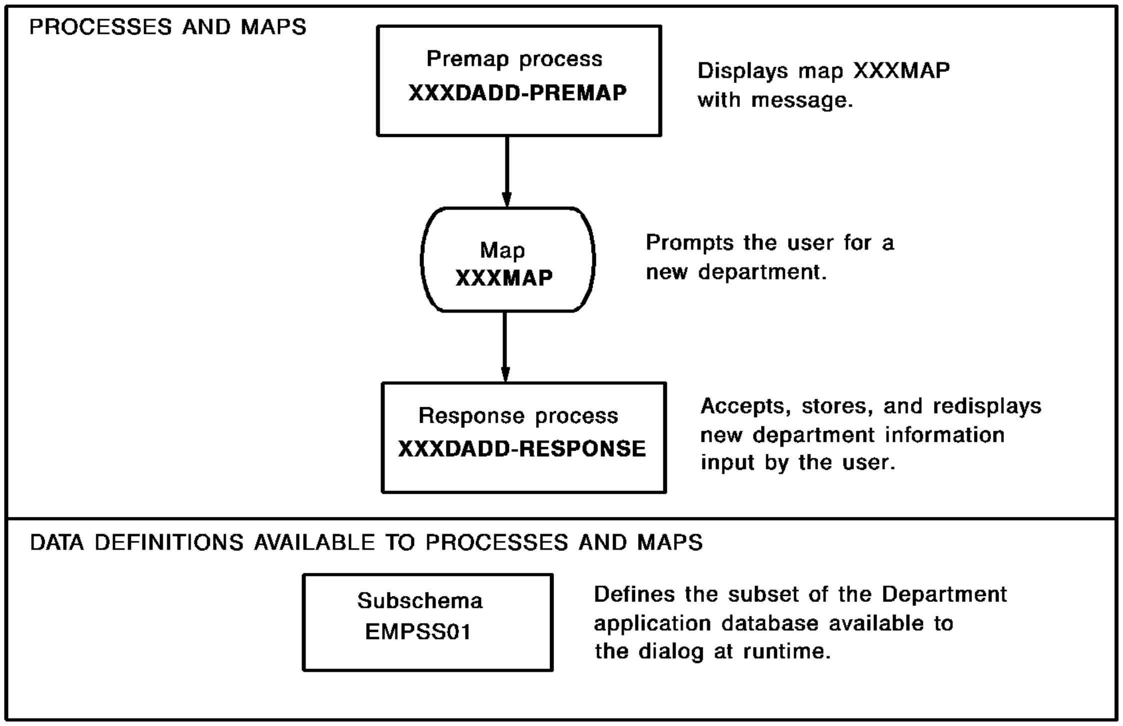
Dialog compiler: When you finish using IDD to define process modules, you use the **CA-ADS dialog compiler (ADSC)** to associate the process modules with dialogs. For example, after you use IDD to define process modules *XXX*DADD-PREMAP and *XXX*DADD-RESPONSE, you use ADSC to associate these process modules with dialog *XXX*DADD.

Enhancing sample application dialogs: In this chapter, you will enhance dialog *XXX*DADD by adding a premap process, response process, and subschema to the dialog, as shown below. Process modules XXXDADD-PREMAP and XXXDADD-RESPONSE perform all processing required for dialog XXXDADD. Subschema EMPSS01 specifies the portion of the database available to process XXXDADD-RESPONSE, which stores department data in the database.

Process name	Type	Function
***XXX*DADD-PREMAP**	Premap	Displays the dialog's map with a message
***XXX*DADD-RESPONSE**	Response	Accepts department information supplied by the end user on the map; stores new department information in the database; redisplays the stored input to the user with a confirming message

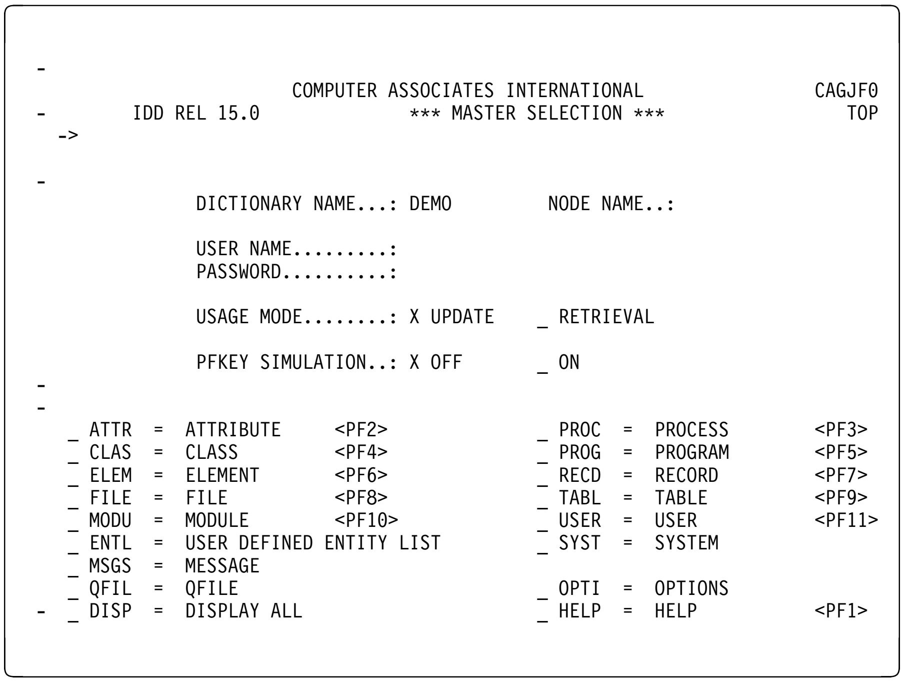
Steps for defining process modules and associating the modules and a subschema with a dialog are discussed below.


A declaration module optionally can be associated with a dialog.
A premap process optionally can be associated with a dialog.
A map optionally can be associated with a dialog.
Any number of response processes optionally can be associated with a dialog.
A subschema must be associated with a dialog uses non-SQL DML statements to access the database.
Any number of SQL-defined tables can be associated with a dialog.
Any number of work records optionally can be associated with a dialog.
PROCESSES AND MAPS
Declaration module
Defines host variables available to the program (SQL programming)
Premap process
Defines processing to be executed before the map is displayed.
Map
Defines the screen displayed by the dialog.
Response process
Response process
One or more response processes define processing that can be executed when the user inputs data on the map.
DATA DEFINITIONS AVAILABLE TO PROCESSES AND MAPS
Subschema
Defines the subset of the non-SQL defined database available to the dialog.
Tables
Defines the SQL-defined tables available to the dialog.
Work record definitions
Defines the work records the dialog can use.


PROCESSES AND MAPS
Premap process
XXXDADD-PREMAP
Displays map XXXMAP with message.
Map
XXXMAP
Prompts the user for a new department.
Response process
XXXDADD-RESPONSE
Accepts, stores, and redisplays new department information input by the user.
DATA DEFINITIONS AVAILABLE TO PROCESSES AND MAPS
Subschema
EMPSS01
Defines the subset of the Department application database available to the dialog at runtime.

12.3 Defining process modules using IDD

You define process modules in the data dictionary. You can associate a given process module with any number of dialogs. You use the **IDD menu facility** to define process modules in the data dictionary. A typical IDD menu facility screen is shown below.



```
-
                       COMPUTER ASSOCIATES INTERNATIONAL                CAGJF0
-       IDD REL 15.0              *** MASTER SELECTION ***                 TOP
  ->

-
              DICTIONARY NAME...: DEMO         NODE NAME..:

              USER NAME.........:
              PASSWORD..........:

              USAGE MODE........: X UPDATE    _ RETRIEVAL

              PFKEY SIMULATION..: X OFF       _ ON
-
-
   _ ATTR  =  ATTRIBUTE     <PF2>             _ PROC  =  PROCESS        <PF3>
   _ CLAS  =  CLASS         <PF4>             _ PROG  =  PROGRAM        <PF5>
   _ ELEM  =  ELEMENT       <PF6>             _ RECD  =  RECORD         <PF7>
   _ FILE  =  FILE          <PF8>             _ TABL  =  TABLE          <PF9>
   _ MODU  =  MODULE        <PF10>            _ USER  =  USER           <PF11>
   _ ENTL  =  USER DEFINED ENTITY LIST        _ SYST  =  SYSTEM
   _ MSGS  =  MESSAGE
   _ QFIL  =  QFILE                           _ OPTI  =  OPTIONS
-  _ DISP  =  DISPLAY ALL                     _ HELP  =  HELP           <PF1>
```

- **Heading and system message area** — Names the IDD screen and returns messages to you.
- **Command area** — Allows you to enter commands and IDD screen names.
- **Specification area** — Prompts you for specifications.
- **Activity selection area** — Allows you to select the next IDD activity from a list of available activities.

If you need any more information on IDD while using the IDD menu facility, see B.6, “Using the IDD Menu Facility” on page B-31.

Steps: To define process modules *XXX*DADD-PREMAP and *XXX*DADD-RESPONSE, you will perform the following steps:

1. Invoke the IDD menu facility.
2. Define process module *XXX*DADD-PREMAP.
3. Define process module *XXX*DADD-RESPONSE.
4. Exit from the IDD menu facility.

These steps are described below. Steps for adding these modules to dialog *XXX*DADD are presented later in this chapter.

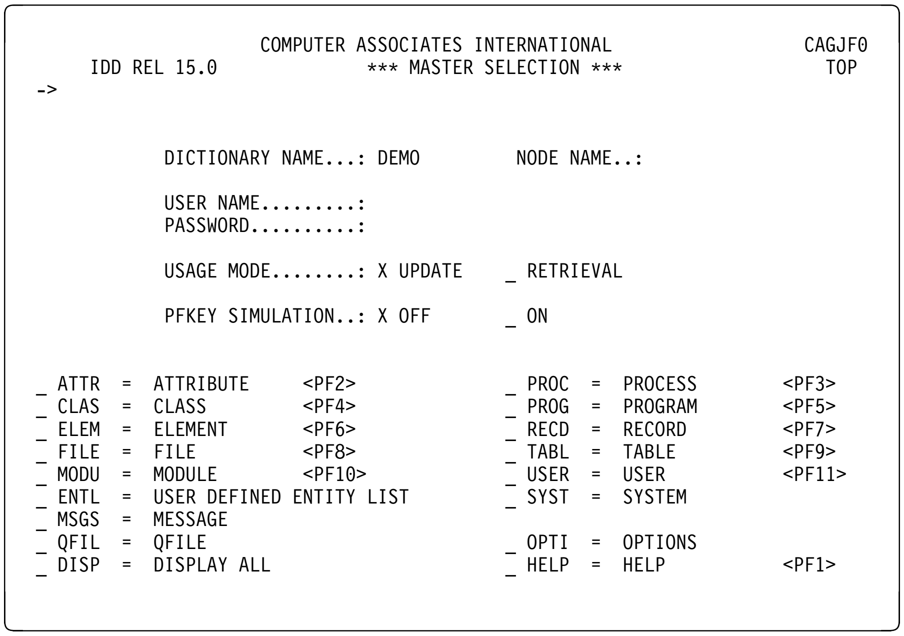
12.3.1 Step 1: Invoke the IDD menu facility

You can invoke IDD from DC/UCF by entering the task code for the IDD menu facility (for example, IDDMT) in response to the prompt presented by DC/UCF. For example, you can invoke IDD from CA-IDMS/DC as shown:

```
ENTER NEXT TASK CODE:
iddmt
                                                      [Enter]
```

For more information on task codes for CA-ADS development tools, see 6.3, "Application development tools" on page 6-7.

Master Selection screen: IDD begins by displaying the Master Selection screen:



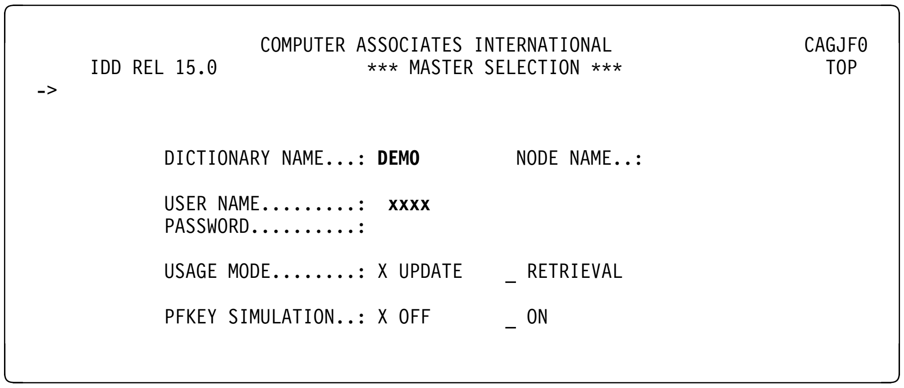
```
                    COMPUTER ASSOCIATES INTERNATIONAL                  CAGJF0
     IDD REL 15.0              *** MASTER SELECTION ***                  TOP
->


            DICTIONARY NAME...: DEMO         NODE NAME..:

            USER NAME.........:
            PASSWORD..........:

            USAGE MODE........: X UPDATE    _ RETRIEVAL

            PFKEY SIMULATION..: X OFF       _ ON


_ ATTR  =  ATTRIBUTE     <PF2>              _ PROC  =  PROCESS        <PF3>
_ CLAS  =  CLASS         <PF4>              _ PROG  =  PROGRAM        <PF5>
_ ELEM  =  ELEMENT       <PF6>              _ RECD  =  RECORD         <PF7>
_ FILE  =  FILE          <PF8>              _ TABL  =  TABLE          <PF9>
_ MODU  =  MODULE        <PF10>             _ USER  =  USER           <PF11>
_ ENTL  =  USER DEFINED ENTITY LIST         _ SYST  =  SYSTEM
_ MSGS  =  MESSAGE
_ QFIL  =  QFILE                            _ OPTI  =  OPTIONS
_ DISP  =  DISPLAY ALL                      _ HELP  =  HELP           <PF1>
```

Screen prompts: When you start an IDD menu facility session, you may need to sign on to IDD. To do so, supply information after one or more of the following Master Selection screen prompts, as appropriate:

- **DICTIONARY NAME** — You must specify the same dictionary, if any, as you specified for all other Department application components. The correct dictionary name may already be displayed in this field.
- **NODE NAME** — You must specify the same dictionary node, if any, as you specified for all other Department application components. The correct dictionary node may already be displayed in this field.

- **USER NAME** — You may need to supply your user ID after this prompt. You can check with others at your site to see if you are required to sign on to the IDD menu facility. If you have a user ID, you can go ahead and enter it, just in case.
- **PASSWORD** — If you need to supply signon information for IDD, you also may need to enter your password after this prompt. Your password is not displayed on the screen when you type it. If you have a password, you can go ahead and enter it.

Signing on: You can sign on to the IDD menu facility as shown:



```
                   COMPUTER ASSOCIATES INTERNATIONAL                  CAGJF0
     IDD REL 15.0              *** MASTER SELECTION ***                  TOP
 ->


           DICTIONARY NAME...: DEMO         NODE NAME..:

           USER NAME.........:  xxxx
           PASSWORD..........:

           USAGE MODE........: X UPDATE     _ RETRIEVAL

           PFKEY SIMULATION..: X OFF        _ ON
```

When you are successfully signed on to the IDD menu facility, the following message is displayed on the Master Selection screen:

```
SIGNON TO IDD WAS SUCCESSFUL
```

From the Master Selection screen, you can display the Process Entity screen and define process module *XXX*DADD-PREMAP, as described in Step 2.

12.3.2 Step 2: Define process module XXXDADD-PREMAP

In this step, you will define process module *XXX*DADD-PREMAP. This process module displays the dialog's map with a message. This processing is performed by the DISPLAY command, as shown below.

```
DISPLAY MSG TEXT
    'ENTER DEPARTMENT INFORMATION, OR SELECT: MOD, BACK, OR EXIT'.
```

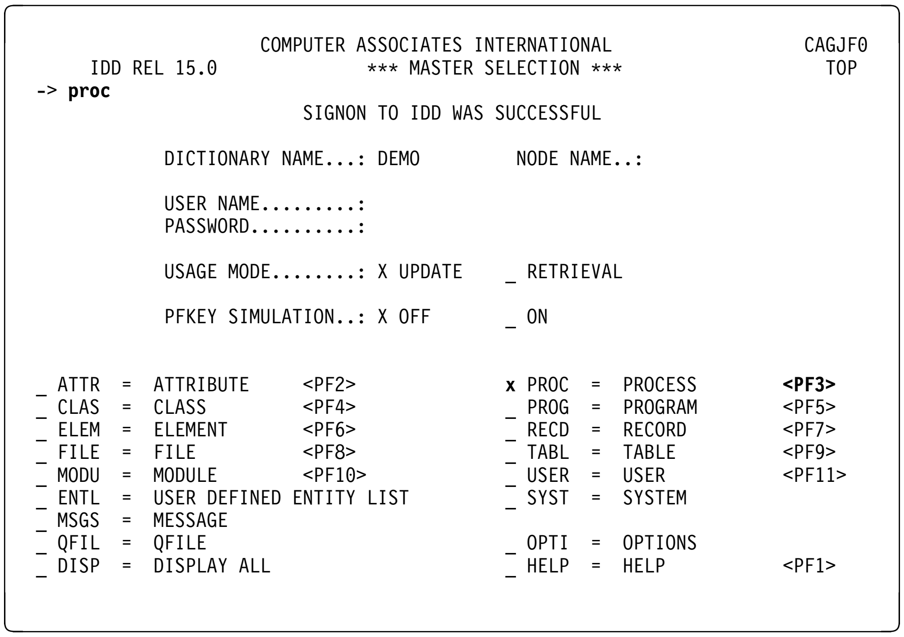
The map is displayed the message that is defined here between single quotation marks. In this example, the DISPLAY command is entered on two lines and is ended by a period (.).

You use the following IDD menu facility screens to define process modules:

1. Use the **Process Entity screen** to specify basic information about the process module, including the module's name.
2. Use the **Process Source screen** to enter source commands for the process module.

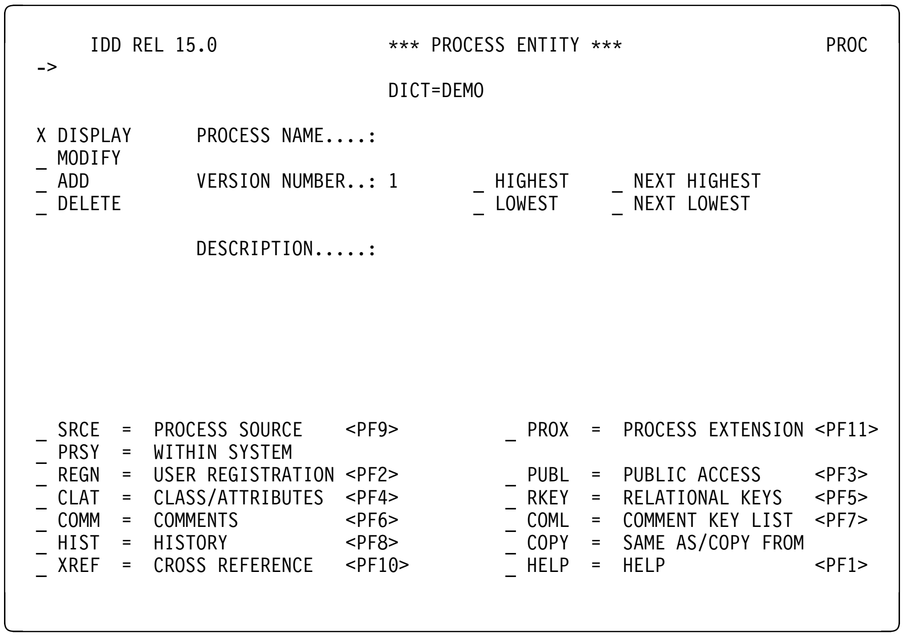
Process Entity screen: To begin defining process module *XXX*DADD-PREMAP, you display the Process Entity screen in any of the following ways:

- Type the identifier for the screen in the command area and press [Enter].
- Type a nonblank character in front of a screen identifier and press [Enter].
- Press a control key to display the associated screen.



```
                    COMPUTER ASSOCIATES INTERNATIONAL                  CAGJF0
     IDD REL 15.0              *** MASTER SELECTION ***                  TOP
-> proc
                        SIGNON TO IDD WAS SUCCESSFUL

            DICTIONARY NAME...: DEMO         NODE NAME..:

            USER NAME.........:
            PASSWORD..........:

            USAGE MODE........: X UPDATE    _ RETRIEVAL

            PFKEY SIMULATION..: X OFF       _ ON


_ ATTR  =  ATTRIBUTE     <PF2>              x PROC  =  PROCESS        <PF3>
_ CLAS  =  CLASS         <PF4>              _ PROG  =  PROGRAM        <PF5>
_ ELEM  =  ELEMENT       <PF6>              _ RECD  =  RECORD         <PF7>
_ FILE  =  FILE          <PF8>              _ TABL  =  TABLE          <PF9>
_ MODU  =  MODULE        <PF10>             _ USER  =  USER           <PF11>
_ ENTL  =  USER DEFINED ENTITY LIST         _ SYST  =  SYSTEM
_ MSGS  =  MESSAGE
_ QFIL  =  QFILE                            _ OPTI  =  OPTIONS
_ DISP  =  DISPLAY ALL                      _ HELP  =  HELP           <PF1>
```

Process Entity screen: The **Process Entity** screen is displayed.



```
     IDD REL 15.0                *** PROCESS ENTITY ***                  PROC
->
                                DICT=DEMO

X DISPLAY      PROCESS NAME....:
_ MODIFY
_ ADD          VERSION NUMBER..: 1        _ HIGHEST    _ NEXT HIGHEST
_ DELETE                                  _ LOWEST     _ NEXT LOWEST

               DESCRIPTION.....:






_ SRCE  =  PROCESS SOURCE    <PF9>          _ PROX  =  PROCESS EXTENSION <PF11>
_ PRSY  =  WITHIN SYSTEM
_ REGN  =  USER REGISTRATION <PF2>          _ PUBL  =  PUBLIC ACCESS     <PF3>
_ CLAT  =  CLASS/ATTRIBUTES  <PF4>          _ RKEY  =  RELATIONAL KEYS   <PF5>
_ COMM  =  COMMENTS          <PF6>          _ COML  =  COMMENT KEY LIST  <PF7>
_ HIST  =  HISTORY           <PF8>          _ COPY  =  SAME AS/COPY FROM
_ XREF  =  CROSS REFERENCE   <PF10>         _ HELP  =  HELP              <PF1>
```

Screen prompts: When you define a process module, you usually specify information for the following Process Entity screen prompts:

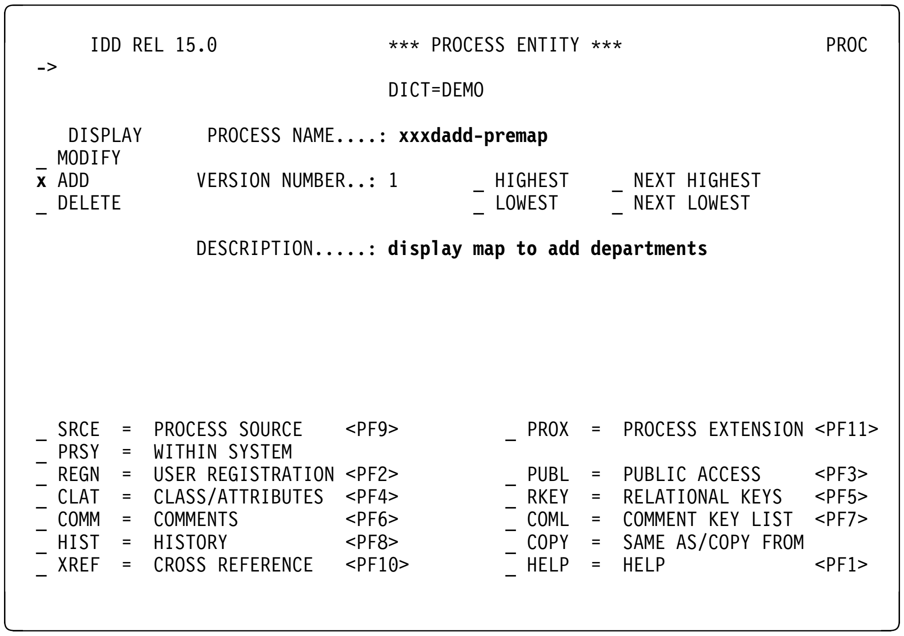
- **PROCESS NAME** — You must supply a process module name. The name that you specify must be unique.
- **DISPLAY** — You deselect the DISPLAY action when you intend to add a new process module.

  To do this, type a blank over the X displayed to the left of the action.
- **ADD** — You select the ADD action to specify that you are defining a new process module.

  To do this, type a nonblank character to the left of the action.
- **DESCRIPTION** — You optionally type a brief description of the process module.

Defining the XXXDADD-PREMAP process module: You define the above basic information for a process module by using the Process Entity screen. For example, you define process module *XXX*DADD-PREMAP by:

- Deselecting the DISPLAY action by typing a space over the X to its left.
- Typing the name for the process module. (You can use your initials instead of *XXX*.)
- Selecting the ADD action.
- Optionally typing a description of the process.



```
    IDD REL 15.0                *** PROCESS ENTITY ***                  PROC
->
                                DICT=DEMO

   DISPLAY      PROCESS NAME....: xxxdadd-premap
_ MODIFY
x ADD          VERSION NUMBER..: 1       _ HIGHEST     _ NEXT HIGHEST
_ DELETE                                 _ LOWEST      _ NEXT LOWEST

               DESCRIPTION.....: display map to add departments





_ SRCE  =  PROCESS SOURCE    <PF9>          _ PROX  =  PROCESS EXTENSION <PF11>
_ PRSY  =  WITHIN SYSTEM
_ REGN  =  USER REGISTRATION <PF2>          _ PUBL  =  PUBLIC ACCESS     <PF3>
_ CLAT  =  CLASS/ATTRIBUTES  <PF4>          _ RKEY  =  RELATIONAL KEYS   <PF5>
_ COMM  =  COMMENTS          <PF6>          _ COML  =  COMMENT KEY LIST  <PF7>
_ HIST  =  HISTORY           <PF8>          _ COPY  =  SAME AS/COPY FROM
_ XREF  =  CROSS REFERENCE   <PF10>         _ HELP  =  HELP              <PF1>
```

When you press [Enter], IDD redisplays the Process Entity screen with a message:

- **If the process module is successfully defined**, the Process Entity screen displays a message like:

  ```
  PROCESS 'XXXDADD-PREMAP' VERSION 1 ADDED
  ```

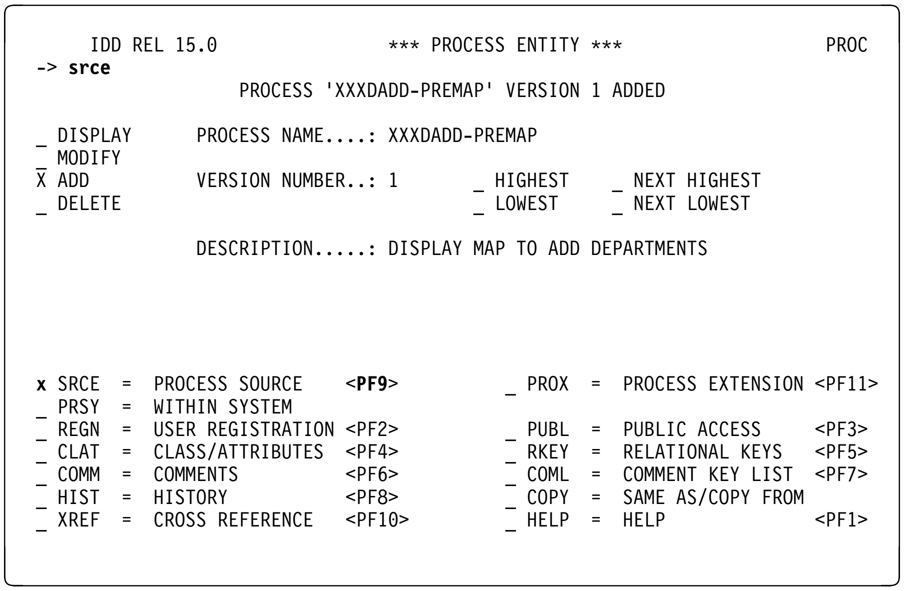
- **If the process module cannot be defined**, the Process Entity screen displays a different message than the message indicated above.

  Read the message to determine the problem. You can type over any errors, and then press [Enter] again.

Entering source statements: After you specify basic information about a process module, you use the **Process Source** screen to enter process commands for the process module.

To display the Process Source screen:

- Type the identifier for the screen in the command area and press [Enter].
- Type a nonblank character in front of a screen identifier and press [Enter].
- Press a control key to display the associated screen.



```
    IDD REL 15.0                *** PROCESS ENTITY ***                  PROC
-> srce
                 PROCESS 'XXXDADD-PREMAP' VERSION 1 ADDED

_ DISPLAY      PROCESS NAME....: XXXDADD-PREMAP
_ MODIFY
X ADD          VERSION NUMBER..: 1        _ HIGHEST    _ NEXT HIGHEST
_ DELETE                                  _ LOWEST     _ NEXT LOWEST

               DESCRIPTION.....: DISPLAY MAP TO ADD DEPARTMENTS




x SRCE  =  PROCESS SOURCE    <PF9>         _ PROX  =  PROCESS EXTENSION <PF11>
_ PRSY  =  WITHIN SYSTEM
_ REGN  =  USER REGISTRATION <PF2>         _ PUBL  =  PUBLIC ACCESS     <PF3>
_ CLAT  =  CLASS/ATTRIBUTES  <PF4>         _ RKEY  =  RELATIONAL KEYS   <PF5>
_ COMM  =  COMMENTS          <PF6>         _ COML  =  COMMENT KEY LIST  <PF7>
_ HIST  =  HISTORY           <PF8>         _ COPY  =  SAME AS/COPY FROM
_ XREF  =  CROSS REFERENCE   <PF10>        _ HELP  =  HELP              <PF1>
```

Entering process statements: Enter the process statements in the screen's text entry area.

Caution: Don't type any characters beyond column 72.

```
     IDD REL 15.0           *** PROCESS SOURCE ***                        SRCE
 ->                                    NO DATA LINES CURRENTLY EXIST
                        PROCESS 'XXXDADD-PREMAP' VERSION 1

 ---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----
display msg text
    'enter department information, or select: mod, back, or exit'.
```

Considerations

- Enclose the message text in single quotation marks.
- End the DISPLAY command with a period.

After pressing [Enter], the Process Source screen is redisplayed. Look over the redisplayed process statement for possible syntax errors in the process code.

Note: IDD does not compile the process code; the code is compiled when the process module is associated with a dialog under ADSC.

```
     IDD REL 15.0           *** PROCESS SOURCE ***                        SRCE
 ->                                    PAGE 1 LINE 1                 1/2
                    PROCESS 'XXXDADD-PREMAP' VERSION 1 MODIFIED

 ---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----
DISPLAY MSG TEXT
    'ENTER DEPARTMENT INFORMATION, OR SELECT: MOD, BACK, OR EXIT'.
```

After you press [Enter], IDD adds the process module to the data dictionary and redisplays the Process Source screen with a message like:

```
PROCESS 'XXXDADD-PREMAP' VERSION 1 MODIFIED
```

Syntax errors: It is a good idea to look over the redisplayed commands for syntax errors. This is because your syntax does not get compiled until you add the process module to a dialog by using ADSC. Typical **syntax errors** include:

- Process statements that extend beyond column 72
- Omitted keywords
- Misspelled comments or record element names
- Omitted periods
- Omitted or misplaced single quotation marks
- Single quotation marks entered as double quotation marks

- Double quotation marks entered as single quotation marks within a quoted string

If you make any errors on the Process Source screen, you can type over them and press [Enter] again.

After you finish using the Process Source screen for process module *XXX*DADD-PREMAP, you can proceed to define process module *XXX*DADD-RESPONSE.

12.3.3 Step 3: Define process module XXXDADD-RESPONSE

Process module *XXX*DADD-RESPONSE processes data entered by the user. To add a department record to the database, source commands for process module *XXX*DADD-RESPONSE need to:

1. **Verify that the department is new**. To do this, the process module must determine that the unique department ID specified by the user does not already exist in the database.
2. **Add the department record to the database** if it is a new record.
3. **Redisplay the screen with a confirming message** and allow the user to add another department record.

Additionally, the process module needs to be able to transfer the user to another dialog function (for example, to MODDEP) when requested by the user.

Commands for XXXDADD- RESPONSE: Below are the sample commands for process module *XXX*DADD- RESPONSE. These commands handle all of the above processing. Notice that CA-ADS does not require you to code MOVE statements for data in this case because the record displayed on the screen is the database record itself. Also notice that one STORE command stores the entire Department record in the database. You define a message for display to the user by including the message in the DISPLAY command. In this example, the DISPLAY command is entered on two lines and is ended by a period (.). This process module evaluates data input by the user and stores new department records in the database.

```
READY USAGE-MODE UPDATE.                              ┐
IF AGR-CURRENT-RESPONSE NE SPACES                     │
AND FIELD DEPT-ID-0410 NOT CHANGED                    │  1
THEN                                                  │
EXECUTE NEXT FUNCTION.                                ┘

OBTAIN CALC DEPARTMENT.                               ┐
IF DB-REC-NOT-FOUND                                   │  2
THEN DO.                                              ┘

STORE DEPARTMENT.                                     ┐
    DISPLAY MSG TEXT                                  │
'DEPARTMENT ADDED'.                                   │  3
END.                                                  ┘

DISPLAY MSG TEXT                                      ┐  4
'TRY AGAIN, OR SELECT: MOD, BACK, OR EXIT'.           ┘
```

1 If the user enters a valid response name (such as MOD) in the map's $RESPONSE field *and* doesn't try to add a new department, the runtime system terminates this response process and transfers execution to the function (such as MODDEP) requested by the user. (The runtime system stores valid responses in system field AGR-CURRENT-RESPONSE.)

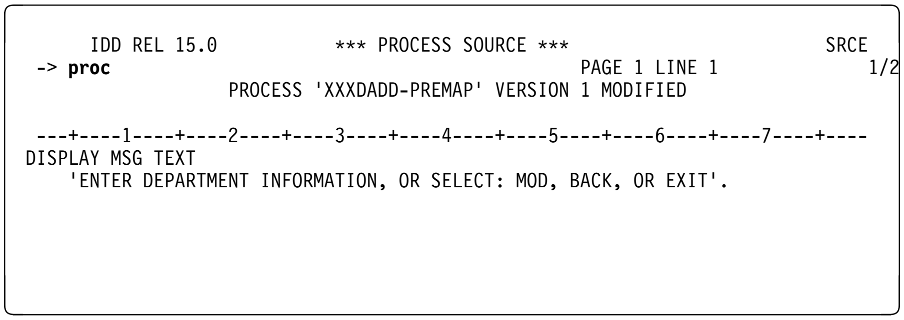
2 This statement attempts to locate the specified department in the database.

3 If the specified department is not in the database, these statements store the department in the database and then redisplay the screen with the DEPARTMENT ADDED message.

4 If the specified department already exists in the database, this statement redisplays the screen with the TRY AGAIN error message.

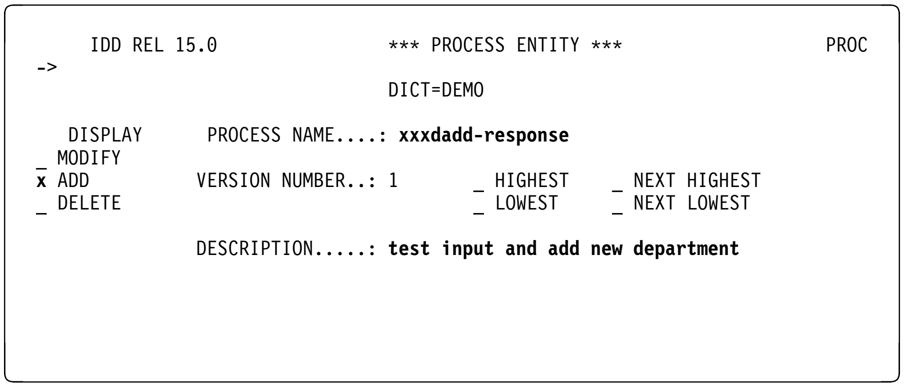
Specifying basic information for the process module: You start defining process module *XXX*DADD-RESPONSE by using the IDD **Process Entity** screen to define basic information for the process module.

Display a blank Process Entity screen.



```
     IDD REL 15.0          *** PROCESS SOURCE ***                      SRCE
 -> proc                                             PAGE 1 LINE 1            1/2
                 PROCESS 'XXXDADD-PREMAP' VERSION 1 MODIFIED

 ---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----
DISPLAY MSG TEXT
    'ENTER DEPARTMENT INFORMATION, OR SELECT: MOD, BACK, OR EXIT'.
```

Process Entity screen: The Process Entity screen is displayed.



```
     IDD REL 15.0                *** PROCESS ENTITY ***                  PROC
->
                                 DICT=DEMO

   DISPLAY      PROCESS NAME....: xxxdadd-response
_ MODIFY
x ADD          VERSION NUMBER..: 1       _ HIGHEST    _ NEXT HIGHEST
_ DELETE                                 _ LOWEST     _ NEXT LOWEST

               DESCRIPTION.....: test input and add new department
```

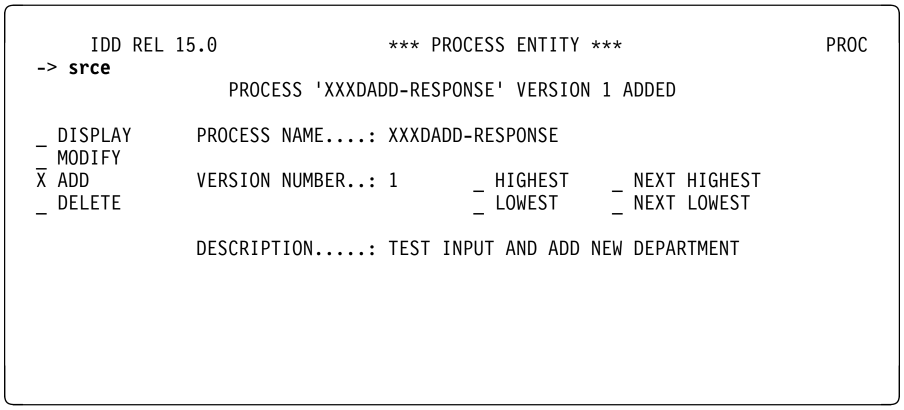
After you press [Enter], the Process Entity screen is redisplayed with a message like:

```
PROCESS 'XXXDADD-RESPONSE' VERSION 1 ADDED
```

This message indicates that your process module specifications have been added to the data dictionary.

To continue defining process module *XXX*DADD-RESPONSE, you use the **Process Source screen** to enter source commands.

Display the Process Source screen



```
     IDD REL 15.0                *** PROCESS ENTITY ***                  PROC
-> srce
                  PROCESS 'XXXDADD-RESPONSE' VERSION 1 ADDED

_ DISPLAY      PROCESS NAME....: XXXDADD-RESPONSE
_ MODIFY
X ADD          VERSION NUMBER..: 1       _ HIGHEST    _ NEXT HIGHEST
_ DELETE                                 _ LOWEST     _ NEXT LOWEST

               DESCRIPTION.....: TEST INPUT AND ADD NEW DEPARTMENT
```

Enter process statements on the Process Source screen.

Considerations

- Periods and single quotation marks shown on this screen are required.
- Don't extend statements beyond column 72.
- You can enter blank lines in your source to improve readability.

- You can type spaces to indent lines, making source statements easier to read and debug.

```
      IDD REL 15.0           *** PROCESS SOURCE ***                        SRCE
 ->                                             PAGE 1 LINE 1              1/3
                      PROCESS 'XXXDADD-RESPONSE' VERSION 1

 ---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----
ready usage-mode update.
if agr-current-response ne spaces
and field dept-id-0410 not changed
then
    execute next function.

obtain calc department.
if db-rec-not-found
then do.

    store department.
    display msg text
        'department added'.
end.

display msg text
    'try again, or select: mod, back, or exit'.
```

After you press [Enter], the Process Source screen is redisplayed. Look over the redisplayed process statements for possible syntax errors. Make sure that the period is outside the single quote on the message.

```
      IDD REL 15.0           *** PROCESS SOURCE ***                        SRCE
 ->                                             PAGE 1 LINE 1             1/14
                  PROCESS 'XXXDADD-RESPONSE' VERSION 1 MODIFIED

 ---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----
READY USAGE-MODE UPDATE.
IF AGR-CURRENT-RESPONSE NE SPACES
AND FIELD DEPT-ID-0410 NOT CHANGED
THEN
EXECUTE NEXT FUNCTION.
OBTAIN CALC DEPARTMENT.
IF DB-REC-NOT-FOUND
THEN DO.
STORE DEPARTMENT.
    DISPLAY MSG TEXT
'DEPARTMENT ADDED'.
END.
DISPLAY MSG TEXT
'TRY AGAIN, OR SELECT: MOD, BACK, OR EXIT'.
```

After you press [Enter], IDD adds the process module to the data dictionary and redisplays the Process Source screen with a message like:

```
PROCESS 'XXXDADD-RESPONSE' VERSION 1 MODIFIED
```

This message indicates that the entry for your process module in the data dictionary has successfully been modified.

If you notice any errors on the redisplayed screen, you can type over the errors to correct them, and then press [Enter] again.

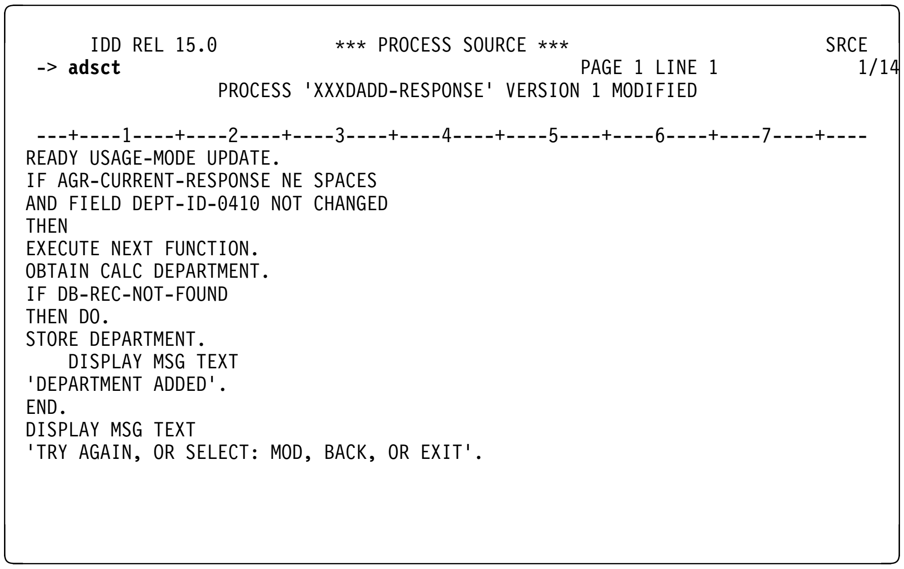
After you finish defining process module *XXX*DADD-RESPONSE, you can exit from IDD.

12.3.4 Step 4: Exit from IDD

When you are using the IDD menu facility under the transfer control facility (TCF), you can exit from IDD by using the SWITCH command.

You can use SWITCH to either transfer to another development tool or return to DC/UCF. In the this sample session, you will transfer to ADSC so that you can associate process modules *XXX*DADD-PREMAP and *XXX*DADD-RESPONSE with dialog *XXX*DADD.

To transfer to ADSC, enter the task code for ADSC (for example, ADSCT) along with the SWITCH command in the **command area** of any IDD menu facility screen:



```
      IDD REL 15.0           *** PROCESS SOURCE ***                        SRCE
 -> adsct                                           PAGE 1 LINE 1             1/14
                PROCESS 'XXXDADD-RESPONSE' VERSION 1 MODIFIED

 ---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----
READY USAGE-MODE UPDATE.
IF AGR-CURRENT-RESPONSE NE SPACES
AND FIELD DEPT-ID-0410 NOT CHANGED
THEN
EXECUTE NEXT FUNCTION.
OBTAIN CALC DEPARTMENT.
IF DB-REC-NOT-FOUND
THEN DO.
STORE DEPARTMENT.
    DISPLAY MSG TEXT
'DEPARTMENT ADDED'.
END.
DISPLAY MSG TEXT
'TRY AGAIN, OR SELECT: MOD, BACK, OR EXIT'.
```

12.4 Adding process modules to dialogs using ADSC

You now will use ADSC to add process modules *XXX*DADD-PREMAP and *XXX*DADD-RESPONSE to dialog *XXX*DADD. You will associate a subschema with the dialog to make a subset of the database available to dialog *XXX*DADD and its process modules. Additionally, you will specify dialog options to help you debug the dialog.

Steps: To enhance dialog *XXX*DADD, you will perform the following steps:

1. Retrieve dialog *XXX*DADD.
2. Specify dialog options for use during development.
3. Add a subschema to the dialog.
4. Add the premap and response processes to the dialog.
5. Recompile the dialog load module.

Steps for viewing and correcting compile-time errors in process modules are discussed later in this chapter, in 12.4.6, “Correct errors in process modules” on page 12-34.

12.4.1 Step 1: Retrieve dialog XXXDADD

In order to retrieve a dialog, you use ADSC.

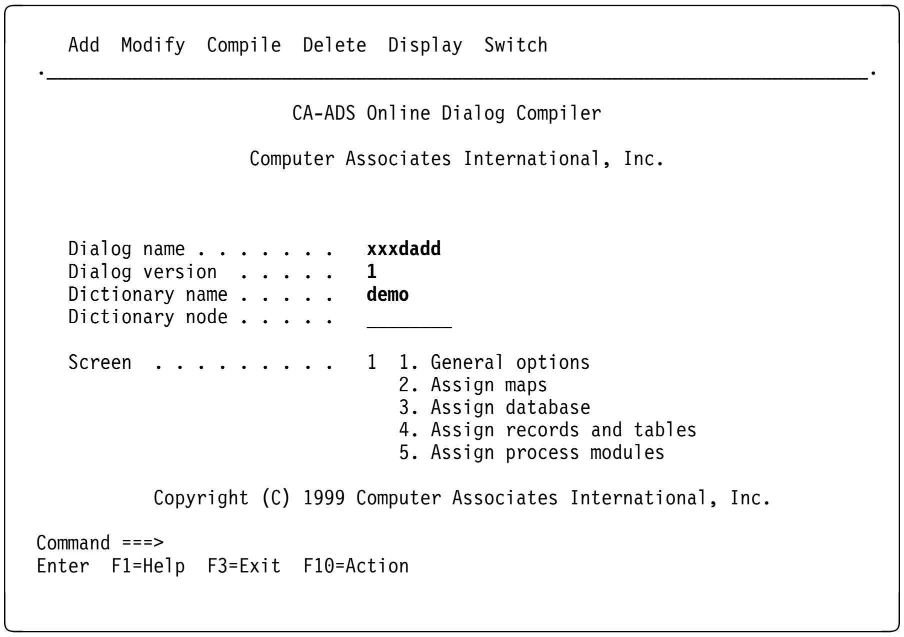
If you *did not* transfer to ADSC earlier in this chapter when you exited from IDD, you need to invoke ADSC by using the task code (for example, ADSCT) for ADSC. For more information on invoking ADSC, see 9.3.1, “Step 1: Invoke ADSC” on page 9-8.

ADSC begins by displaying the **Main Menu** screen. You use a blank Main Menu screen to retrieve a dialog.

Screen prompts: You typically enter information after one or more of the following Main Menu screen prompts:

- **Dialog name** — You must specify the name (*XXX*DADD) that you used when you defined the dialog in 9.3.2, “Step 2: Define dialog XXXDADD” on page 9-9.
- **Dictionary name** — You must specify the same dictionary, if any, as you specified for your dialog in Chapter 4. The correct dictionary name may already be displayed in this field.
- **Dictionary node** — You must specify the same dictionary node, if any, as you specified for your dialog definition. The correct dictionary node may already be displayed in this field.

You use the ADSC Main Menu screen to retrieve dialog *XXX*DADD:



```
   Add  Modify  Compile  Delete  Display  Switch
._____________________________________________________________________________.

                      CA-ADS Online Dialog Compiler

                  Computer Associates International, Inc.



   Dialog name . . . . . . .   xxxdadd
   Dialog version  . . . . .   1
   Dictionary name . . . . .   demo
   Dictionary node . . . . .   ________

   Screen  . . . . . . . . .   1  1. General options
                                  2. Assign maps
                                  3. Assign database
                                  4. Assign records and tables
                                  5. Assign process modules

           Copyright (C) 1999 Computer Associates International, Inc.

Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

Press [Enter] to retrieve the application.

After you press [Enter], ADSC redisplays the Main Menu screen with a message confirming that the dialog is available for modification.

Note: If the dialog has not been explicitly released (using the **Release** option of the **Modify** action on the action bar of the Main Menu), naming the dialog on the Main Menu screen retrieves that definition for modification. If the dialog has been released, you use the ADSC Main Menu screen to check out the dialog definition for modification (using the **Checkout** option of the **Modify** action on the action bar of the Main Menu).

For information on checking out a dialog, see *CA-ADS Reference*.

If the dialog has been released, subsequently checked out to another developer and not released by that developer, you will not be able to check it out.

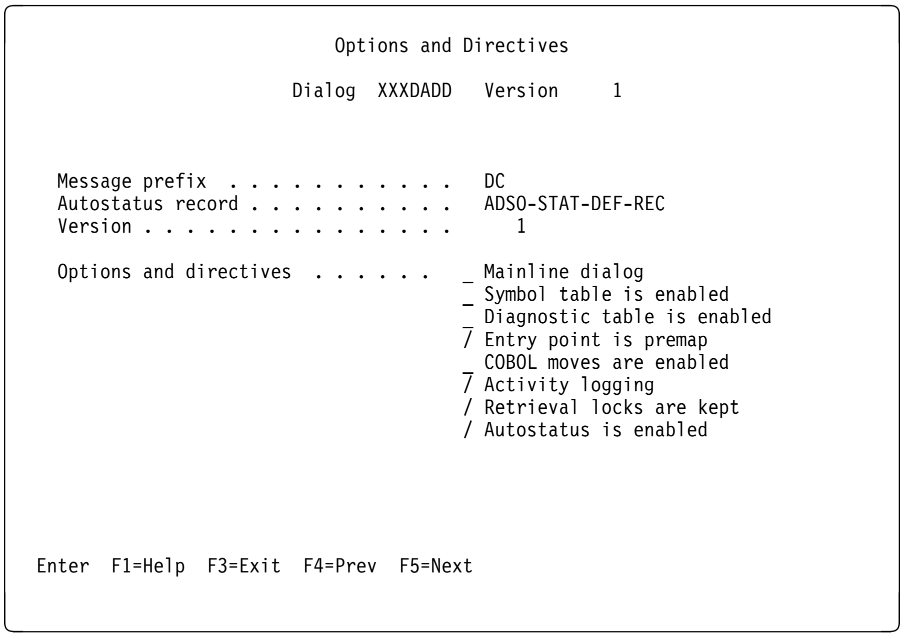
After you press [Enter], ADSC displays dialog *XXX*DADD on the Main Menu screen, with appropriate messages.

You can now specify dialog options on the Options and Directives screen. You access the Options and Directives screen by choosing option **1** from the Main Menu.

12.4.2 Step 2: Specify dialog options

You use the **Options and Directives** screen to specify options to help you develop and debug dialogs. A sample Options and Directives screen is shown below:

Sample Options and Directives screen



```
                              Options and Directives

                         Dialog  XXXDADD   Version     1



     Message prefix  . . . . . . . . . . .   DC
     Autostatus record . . . . . . . . . .   ADSO-STAT-DEF-REC
     Version . . . . . . . . . . . . . . .      1

     Options and directives  . . . . . .   _ Mainline dialog
                                           _ Symbol table is enabled
                                           _ Diagnostic table is enabled
                                           / Entry point is premap
                                           _ COBOL moves are enabled
                                           / Activity logging
                                           / Retrieval locks are kept
                                           / Autostatus is enabled




   Enter  F1=Help  F3=Exit  F4=Prev  F5=Next
```

Screen prompts: During development, you typically use the following Options and Directives screen prompts to enable options for the dialog:

- **Symbol table is enabled** — You can create a symbol table to detect, trace, and resolve programming errors in dialogs.

  The symbol table stores information about the dialog, such as record element names and internal line numbers for commands in process modules. This information allows the **online debugger** to track, set breakpoints in, and alter execution of a dialog.

  >> For more information on using the online debugger with CA-ADS, see *CA-ADS Reference*.

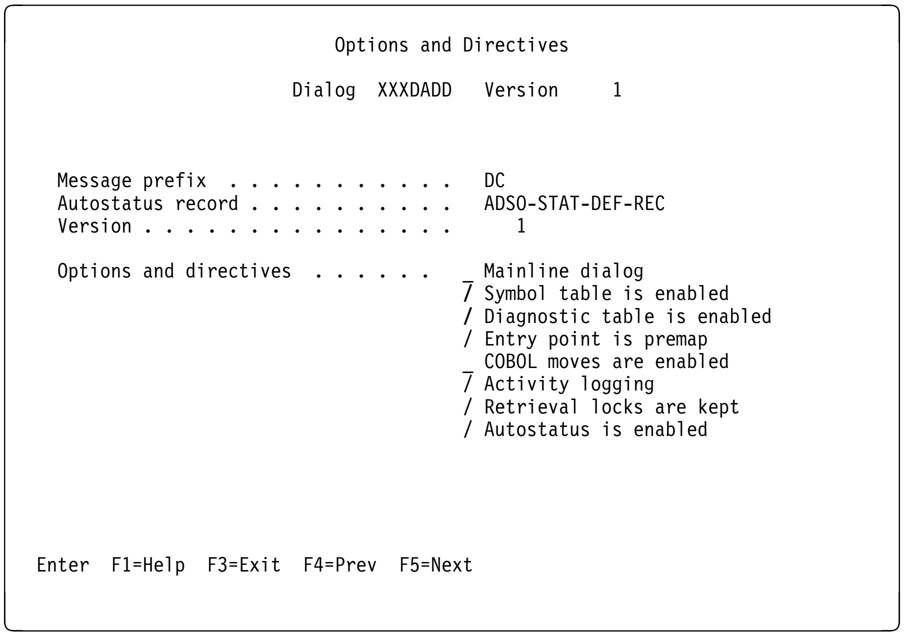
- **Diagnostic table is enabled** — You can create a diagnostic table to help locate the causes for dialog abends during development.

  When a dialog abends, the dialog's diagnostic table allows the **Dialog Abort Information screen** to display the process command that was being executed when the abend occurred.

  >> For more information on diagnostic tables, see *CA-ADS Reference*.

When the application is ready for production use, you can disable dialog symbol and diagnostic tables and disable display of the Dialog Abort Information screen.

Specifying XXXDADD options: To specify development options for dialog *XXX*DADD, display and use the Options and Directives screen. Enable the symbol table and diagnostic table.



```
                        Options and Directives

                    Dialog  XXXDADD   Version     1



   Message prefix  . . . . . . . . . . .   DC
   Autostatus record . . . . . . . . . .   ADSO-STAT-DEF-REC
   Version . . . . . . . . . . . . . . .      1

   Options and directives  . . . . . .   _ Mainline dialog
                                         / Symbol table is enabled
                                         / Diagnostic table is enabled
                                         / Entry point is premap
                                         _ COBOL moves are enabled
                                         / Activity logging
                                         / Retrieval locks are kept
                                         / Autostatus is enabled




 Enter  F1=Help  F3=Exit  F4=Prev  F5=Next
```

After you press [Enter], ADSC displays a confirming message on the Options and Directives screen to inform you that your specifications contain no errors.

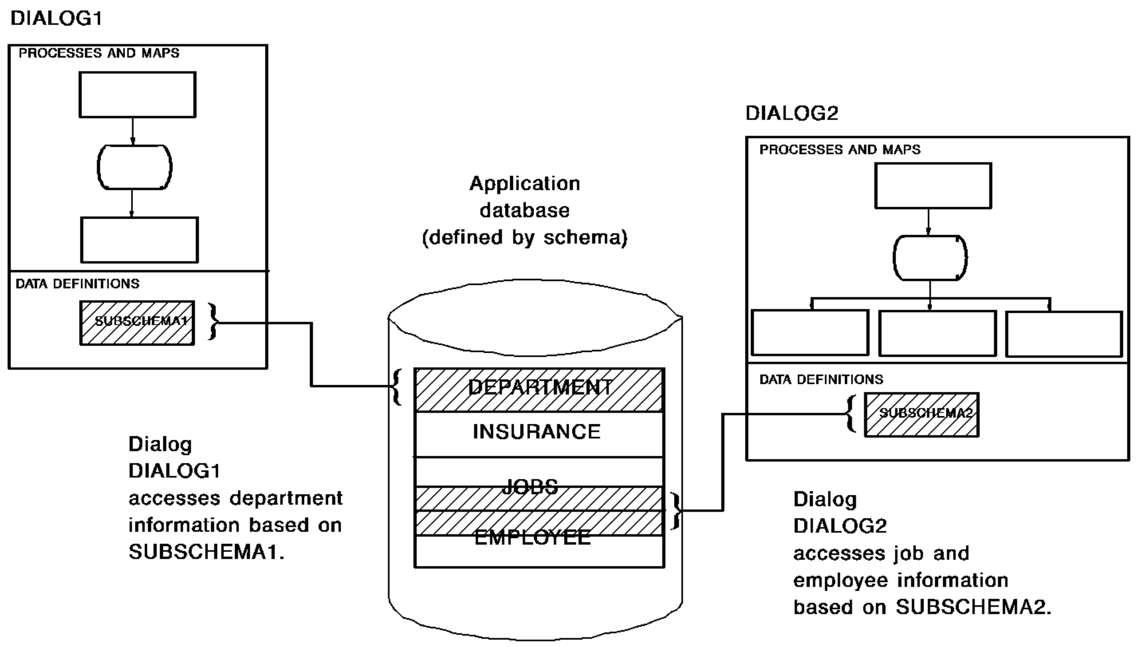
You are now ready to add database information to the dialog definition. Return to the Main Menu by pressing [PF3].

12.4.3 Step 3: Add a subschema

Using non-SQL DML statements to access a database: A non-SQL defined database is defined by a **schema**. A schema typically includes definitions for the database records required by your application. For example, the schema for a fully developed personnel application probably would include records for employee information, job descriptions, and hospital and dental insurance information.

To promote efficient use of the database and other system resources at runtime, you restrict each dialog to a specific subset of the database. Each subset of the database is defined by a **subschema**. The subschema identifies the subset of the database that the dialog can access at runtime.

The way that a subschema relates a dialog to the application's database is shown below. Schemas and subschemas usually are defined by database administrators


DIALOG1
PROCESSES AND MAPS
DATA DEFINITIONS
SUBSCHEMA1
Application database (defined by schema)
DEPARTMENT
INSURANCE
JOBS
EMPLOYEE
DIALOG2
PROCESSES AND MAPS
DATA DEFINITIONS
SUBSCHEMA2
Dialog DIALOG1 accesses department information based on SUBSCHEMA1.
Dialog DIALOG2 accesses job and employee information based on SUBSCHEMA2.

(DBAs) or system administrators, based on application requirements. A subschema determines the portion of the database (schema) available to the dialog at runtime.

You need to associate a subschema with each dialog that accesses a non-SQL defined database using non-SQL DML statements. For example, you will associate a subschema with dialog *XXX*DADD because commands in process module *XXX*DADD-RESPONSE access the database to store new department information using non-SQL DML statements.

You associate the subschema with the dialog before you add process *XXX*DADD-RESPONSE so that ADSC can verify the process module's database commands.

You use the **Database Specifications** screen to associate a subschema with dialog *XXX*DADD.

Using SQL statements to access a database: An SQL-defined application database is defined by tables associated with a schema. A schema typically includes definitions for the tables required by your application. For example, the schema for a fully developed personnel application probably would include tables for employee information, job descriptions, and hospital and dental insurance information.

To promote efficient use of the database and other system resources at runtime in the SQL environment, you identify an **access module** to be associated with the dialog. An access module identifies the method of access that the dialog will use at runtime.

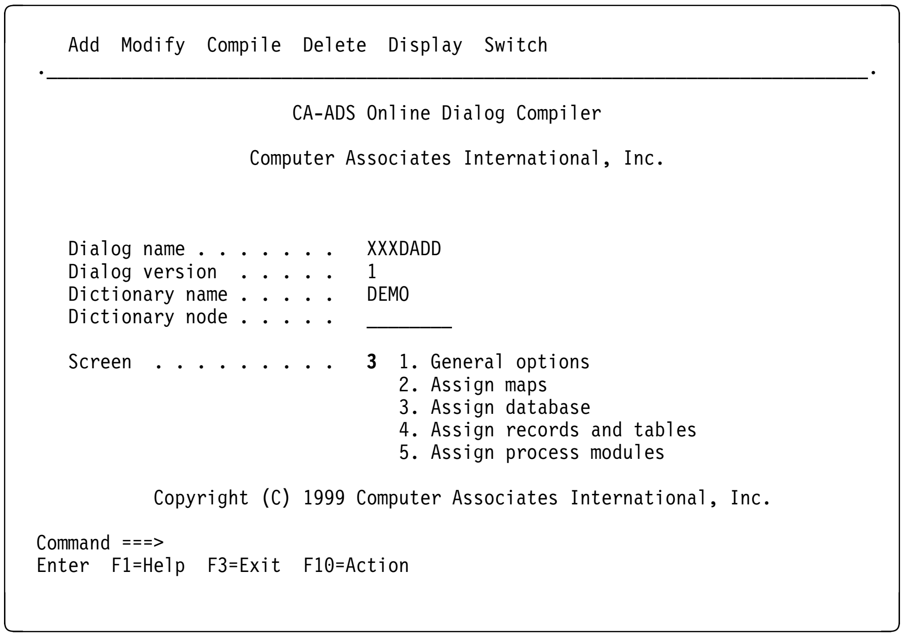
Access modules are made up of relational command modules (RCMs), and are usually created by the application developer.

>> For information on creating an access module, see *CA-IDMS SQL Programming*.

Modifying the sample dialogs: In the *XXX*DADD and *XXX*DUPD dialogs, you are accessing a non-SQL defined database. Therefore, you must associate a subschema with each of these dialogs.

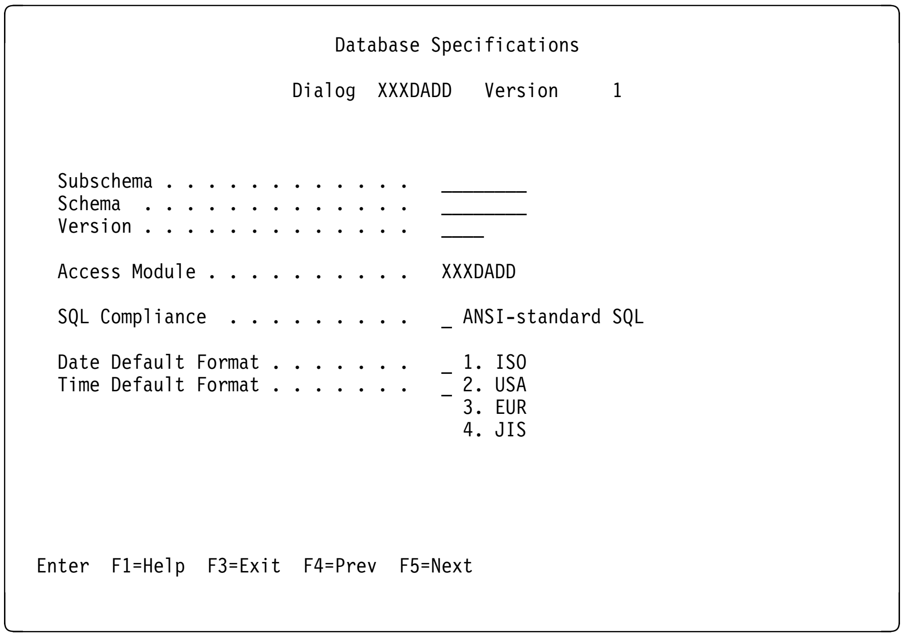
You use the **Database Specifications** screen to associate a subschema with dialog *XXX*DADD.

Accessing the Database Specifications screen: To access the Database Specifications screen, you enter **3** at the **Screen** prompt on the Main Menu.



```
   Add  Modify  Compile  Delete  Display  Switch
 .______________________________________________________________________________.

                       CA-ADS Online Dialog Compiler

                   Computer Associates International, Inc.



   Dialog name . . . . . . .   XXXDADD
   Dialog version  . . . . .   1
   Dictionary name . . . . .   DEMO
   Dictionary node . . . . .   ________

   Screen  . . . . . . . . .   3  1. General options
                                  2. Assign maps
                                  3. Assign database
                                  4. Assign records and tables
                                  5. Assign process modules

           Copyright (C) 1999 Computer Associates International, Inc.

 Command ===>
 Enter  F1=Help  F3=Exit  F10=Action
```

Sample Database Specifications screen: The Database Specifications screen is displayed.



```
                         Database Specifications

                     Dialog  XXXDADD   Version     1



   Subschema . . . . . . . . . . . .   ________
   Schema  . . . . . . . . . . . . .   ________
   Version . . . . . . . . . . . . .   ____

   Access Module . . . . . . . . . .   XXXDADD

   SQL Compliance  . . . . . . . . .   _ ANSI-standard SQL

   Date Default Format . . . . . . .   _ 1. ISO
   Time Default Format . . . . . . .   _ 2. USA
                                         3. EUR
                                         4. JIS




 Enter  F1=Help  F3=Exit  F4=Prev  F5=Next
```

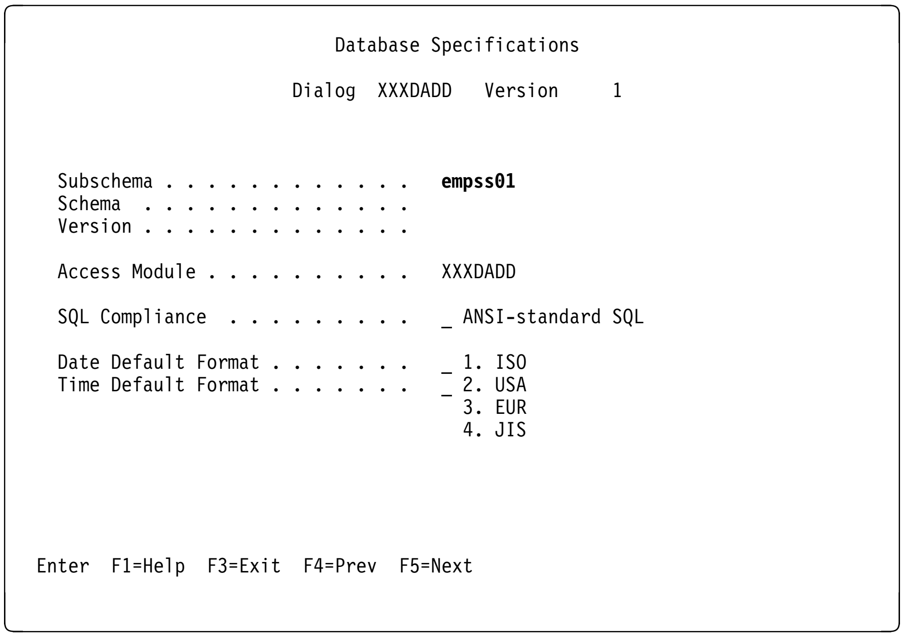
Screen prompts: To associate a subschema with a dialog, you use the following **Database Specifications** screen prompts:

- **Subschema** — You name an existing subschema in response to this prompt. A sample subschema name (EMPSS01) is used in this manual; a different name may be required at your site.
- **Schema** — You may need to name the schema for your application in response to this prompt. Naming a schema usually is required only when your schema is not unique.

  For example, schemas typically are not unique when duplicate, identical development and production databases are defined.

  If you know the name of your schema, go ahead and specify it here.

Modifying XXXDADD dialog: Modify the *XXX*DADD dialog to include the subschema EMPSS01.



```
                          Database Specifications

                      Dialog  XXXDADD   Version     1



   Subschema . . . . . . . . . . . .   empss01
   Schema  . . . . . . . . . . . . .
   Version . . . . . . . . . . . . .

   Access Module . . . . . . . . . .   XXXDADD

   SQL Compliance  . . . . . . . . .   _ ANSI-standard SQL

   Date Default Format . . . . . . .   _ 1. ISO
   Time Default Format . . . . . . .   _ 2. USA
                                         3. EUR
                                         4. JIS




 Enter  F1=Help  F3=Exit  F4=Prev  F5=Next
```

Note: By default, the access module is given the same name as the dialog.

When you press [Enter], ADSC associates the named subschema with the dialog if there are no errors, and then redisplays the screen with a confirming message.

A different message is displayed if ADSC detects any errors.

Read the message to determine the problem. Make sure that you have specified the correct subschema on the Database Specifications screen. If you didn't specify a schema name, ask others at your site whether a schema name is required. After you change information on the screen, press [Enter] again.

If the error persists, verify that you specified the correct version of the subschema.

You are now ready to add process modules to the dialog definition. Return to the Main Menu by pressing [PF3].

12.4.4 Step 4: Add process modules

Premap processes: You add a **premap process** to a dialog so the dialog can perform processing or access database information before the dialog's map is displayed. If a dialog has a premap process, that process is executed as soon as the dialog begins.

You add a premap process to a dialog by using the ADSC **Process Modules** screen.

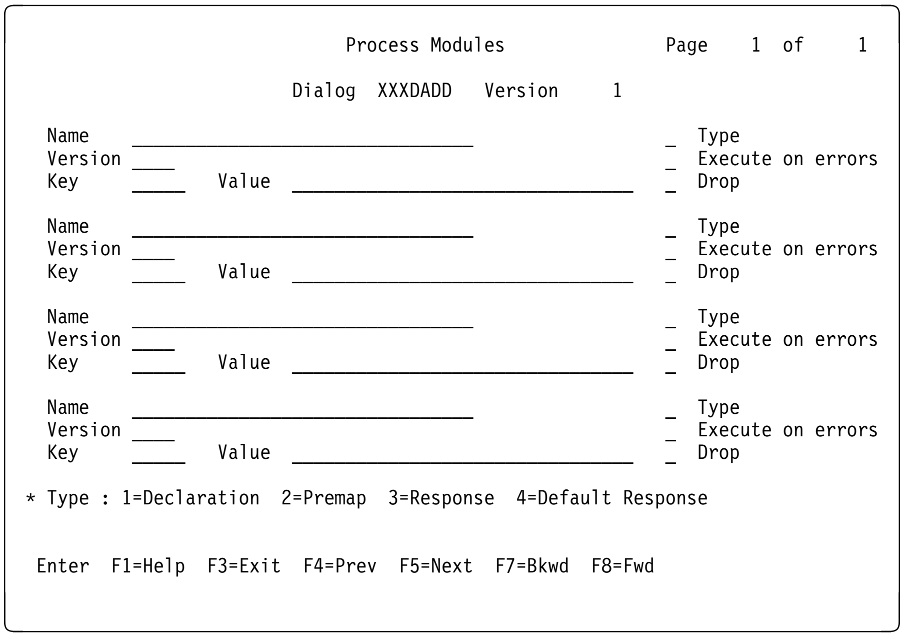
Response processes: A **response process** enables a dialog to perform processing after the user inputs data on the map.

You can add any number of response processes to a dialog, enabling a dialog to perform many different processing operations. At runtime, the response process executed for the dialog is determined by actions taken by the user during dialog execution, as described later in this step.

You add a response process to a dialog by using the ADSC **Process Modules** screen.

Accessing the Process Modules screen: You access the Process Modules screen by entering **5** at the **Screen** prompt on the Main Menu and pressing [Enter].

Sample Process Modules screen: The Process Modules screen is displayed.



```
                             Process Modules              Page    1  of     1

                        Dialog  XXXDADD   Version     1

  Name    ________________________________               _  Type
  Version ____                                           _  Execute on errors
  Key     _____   Value  ________________________________  _  Drop

  Name    ________________________________               _  Type
  Version ____                                           _  Execute on errors
  Key     _____   Value  ________________________________  _  Drop

  Name    ________________________________               _  Type
  Version ____                                           _  Execute on errors
  Key     _____   Value  ________________________________  _  Drop

  Name    ________________________________               _  Type
  Version ____                                           _  Execute on errors
  Key     _____   Value  ________________________________  _  Drop

* Type : 1=Declaration  2=Premap  3=Response  4=Default Response


 Enter  F1=Help  F3=Exit  F4=Prev  F5=Next  F7=Bkwd  F8=Fwd
```

Screen prompts: You specify information about the process modules after prompts on the Process Modules screen:

- **Name** — You specify a process module name after the **Name** prompt. By doing this, you associate the process module with the dialog.
- **Version** — You specify the version number of the associated process module as specified in the dictionary.
- **Type** — You specify the type of process module you have just named.

  1 indicates that this is a **declaration module** used with SQL only.

  >> For information on declaration modules, see *CA-IDMS SQL Programming*.

  2 indicates that this is a **premap process**.

  3 indicates that this is a **response process**.

  4 indicates that this is the **default response process** for the dialog.

At runtime, the dialog's default response process (if any) is executed when the user inputs information on the dialog's map without specifying any response process to be executed.

You will define a premap process and a response process for dialog *XXX*DADD. You will not define a default response process.

- **Key** — If the process module is defined as a response process, you can associate a key (for example, [Enter] or [PF1]) with the process.

  At runtime, the response process is executed for the dialog if the user presses the associated control key while viewing the dialog's map.

  You will associate [Enter] with response process *XXX*DADD-RESPONSE.

- **Value** — If the process module is defined as a response process, you can associate a value (for example, MOD) with the process.

  At runtime, the response process is executed when the dialog's user enters the response field value in the screen's $RESPONSE field.

  For example, assume that you defined response process *XXX*DADD-MOD to be executed when the user requests transfer to function MODDEP. In this case, you would give response process *XXX*DADD-MOD a response field value of MOD. This way, MOD first executes *XXX*DADD-MOD, which then transfers control to function MODDEP.

  You will not add a response field value to response process *XXX*DADD-RESPONSE.

Using ENTER as a key: When you add response process *XXX*DADD-RESPONSE to dialog *XXX*DADD, you will associate the process with [Enter]. ENTER is a good key for this response process because users are accustomed to pressing [Enter] to input information, and are more likely to press [Enter] than a PF key when unfamiliar with the application.

At runtime, whenever the user presses [Enter] to input data for dialog *XXX*DADD, response process *XXX*DADD-RESPONSE is executed.

ENTER is often associated with the response process that performs the dialog's major processing. To make it easier for users to input information, the runtime system *automatically executes* a dialog's ENTER response process when the user inputs information without otherwise specifying a response process to execute.

In the sample application, when using dialog *XXX*DADD, response process *XXX*DADD-RESPONSE is executed if the user:

- Inputs a new department record by pressing [Enter]
- Requests transfer to the MODDEP function by using the MOD response or by pressing [PF2]. This is true in this dialog because:

  1. Response process *XXX*DADD-RESPONSE is associated with [Enter].

2. No response process is associated with MOD or with [PF5], both of which are valid the current function (ADDDEP).

Immediately executable function: As an application developer, you can inhibit execution of *XXX*DADD-RESPONSE when the user requests transfer to MODDEP. To do this, you make function MODDEP an **immediately executable function**.

At runtime, when the user requests transfer to an immediately executable function, control transfers immediately to that function. No response process is executed before transfer occurs.

For more information on immediately executable functions, see the *CA-ADS Reference*.

The EXECUTE NEXT FUNCTION command: In this chapter, you will not make function MODDEP an immediately executable function. Instead, you will enable response process *XXX*DADD-RESPONSE to transfer control. To do this, you include an **EXECUTE NEXT FUNCTION** command in the process module. When executed, this command transfers control to the function specified by the user.

For example, when you defined *XXX*DADD-RESPONSE by using IDD earlier in this chapter, you included the EXECUTE NEXT FUNCTION command in the following conditional structure:

```
IF AGR-CURRENT-RESPONSE NE SPACES
AND DEPT-ID-0415 NOT CHANGED
THEN
    EXECUTE NEXT FUNCTION.       ◄-- Control transfers to the next
                                     function only when the
                                     above two conditions are met.
```

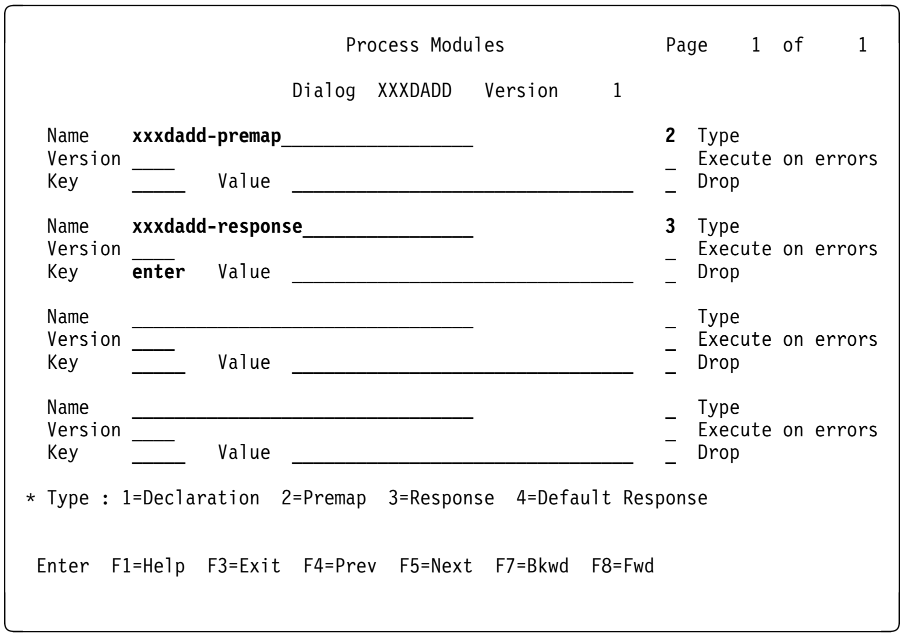
AGR-CURRENT-RESPONSE is an element in the system-supplied ADSO-APPLICATION-GLOBAL-RECORD that the runtime system uses for flow-of-control processing.

This conditional structure causes the following different events to occur at runtime:

- **A department is added and control remains in dialog *XXX*DADD** whenever the user enters new department information.
- **Control transfers** when the user enters a valid response name (AGR-CURRENT-RESPONSE NE SPACES) without entering a new department ID.

Adding process modules to the dialog: You use the Process Modules screen to add two processes to the dialog:

Process module name	**Type**	**Comments**
*XXX*DADD-PREMAP	Premap	
*XXX*DADD-RESPONSE	Response	Key: [Enter]

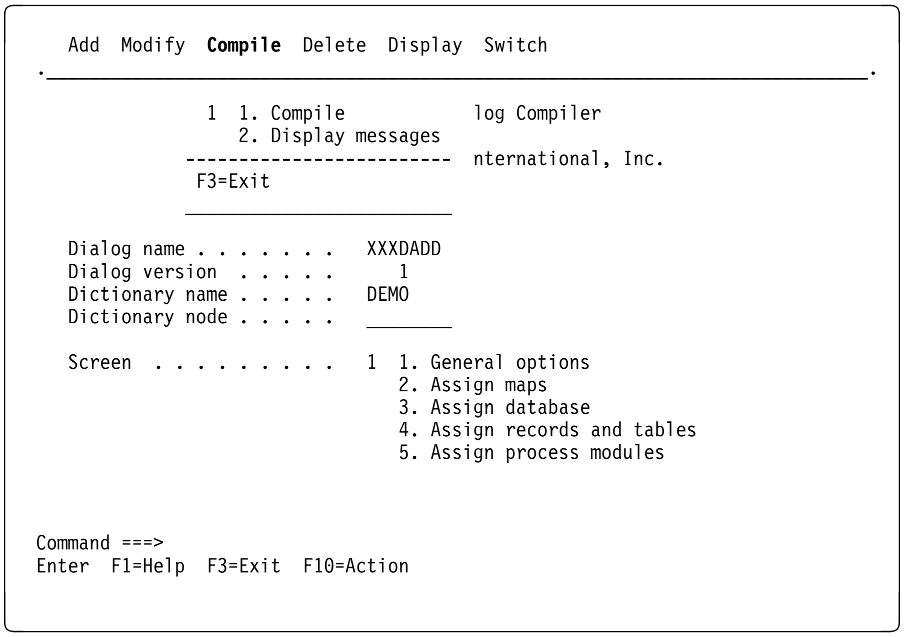


```
                              Process Modules              Page    1  of     1

                         Dialog  XXXDADD   Version     1

  Name    xxxdadd-premap__________________                 2  Type
  Version ____                                             _  Execute on errors
  Key     _____   Value  ________________________________  _  Drop

  Name    xxxdadd-response________________                 3  Type
  Version ____                                             _  Execute on errors
  Key     enter   Value  ________________________________  _  Drop

  Name    ________________________________                 _  Type
  Version ____                                             _  Execute on errors
  Key     _____   Value  ________________________________  _  Drop

  Name    ________________________________                 _  Type
  Version ____                                             _  Execute on errors
  Key     _____   Value  ________________________________  _  Drop

* Type : 1=Declaration  2=Premap  3=Response  4=Default Response


 Enter  F1=Help  F3=Exit  F4=Prev  F5=Next  F7=Bkwd  F8=Fwd
```

ADSC displays a message confirming that the named process module was found in the dictionary. After the processes have been identified, you can recompile the dialog.

12.4.5 Step 5: Recompile the dialog

In this chapter, you have enhanced dialog *XXX*DADD by adding a subschema, premap process, and response process to the dialog definition. To update these modifications to the dialog load module, you must recompile the process modules and recompile the dialog by selecting the compile activity from the action bar on the Main Menu:



```
   Add  Modify  Compile  Delete  Display  Switch
._____________________________________________________________________________.

              1  1. Compile           log Compiler
                 2. Display messages
             -------------------------  nternational, Inc.
              F3=Exit
             _________________________

   Dialog name . . . . . . .   XXXDADD
   Dialog version  . . . . .      1
   Dictionary name . . . . .   DEMO
   Dictionary node . . . . .   ________

   Screen  . . . . . . . . .   1  1. General options
                                  2. Assign maps
                                  3. Assign database
                                  4. Assign records and tables
                                  5. Assign process modules



Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

After you press [Enter] to recompile the dialog, ADSC compiles the process module source code.

ADSC displays a confirming message if no errors are found.

This indicates that the compiled process modules were successfully added to the dialog. If there are no errors, ADSC then creates the dialog load module and redisplays the Main Menu.

Error messages: If ADSC finds errors while compiling a process, it displays an error message.

In this case, you display and correct errors in the process module as discussed in 12.4.6, “Correct errors in process modules” on page 12-34, later in this chapter.

Different messages are displayed depending on the nature of the error.

Read the message to determine the problem. Verify that you have correctly typed the process module name on the Process Modules screen. You can type over errors on the screen, and then press [Enter] again.

After you successfully recompile dialog *XXX*DADD, you can exit from ADSC by pressing [PF3].

Note: When a dialog is added or checked out, a queue is established. The queue is deleted only when:

- The dialog is released with no uncompiled changes
- The dialog is deleted

A dialog with changes that is released can be retrieved by another developer, but the queue remains.

12.4.6 Correct errors in process modules

When you add a premap or response process to a dialog, ADSC compiles the related process module commands. A fully compiled copy of the process module is stored in the dialog.

Compile time errors: When **errors** arise at compile time:

1. ADSC inserts diagnostic messages in the dialog's copy of the process module.
2. ADSC redisplays the Main Menu screen, as appropriate, with a message indicating that there is an error in a particular process module.

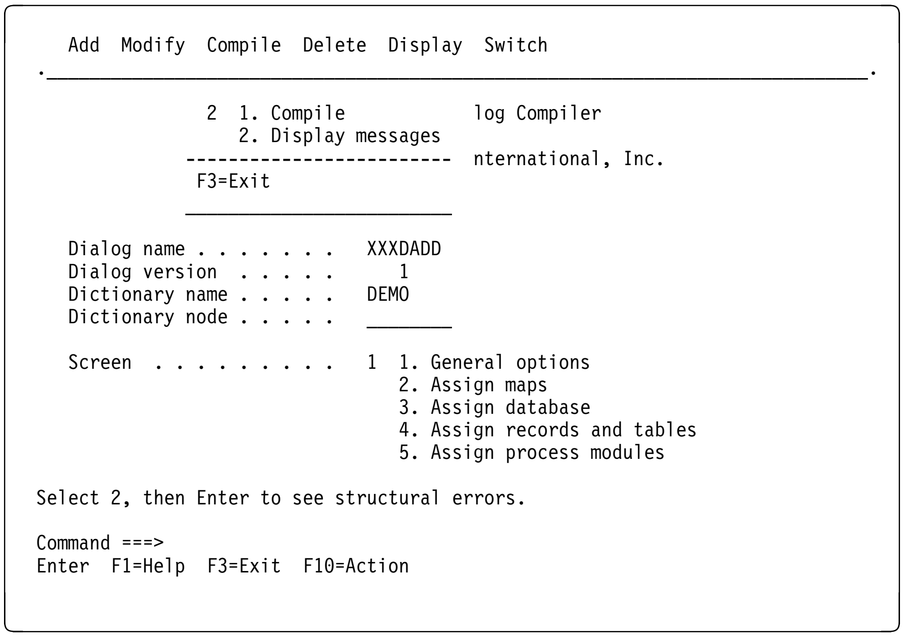
In this case, the process module is *not added* to the dialog. If ADSC indicates that there are compile errors, you must:

1. **Display diagnostic messages** for the process module.
2. **Correct errors**, including:
   - **Discrepancies** between the process module and other dialog components
   - **Syntax errors** in the process module source.
3. **Recompile the process module** and update the process module in the dialog by recompiling the dialog load module.

12.4.6.1 Display structural messages

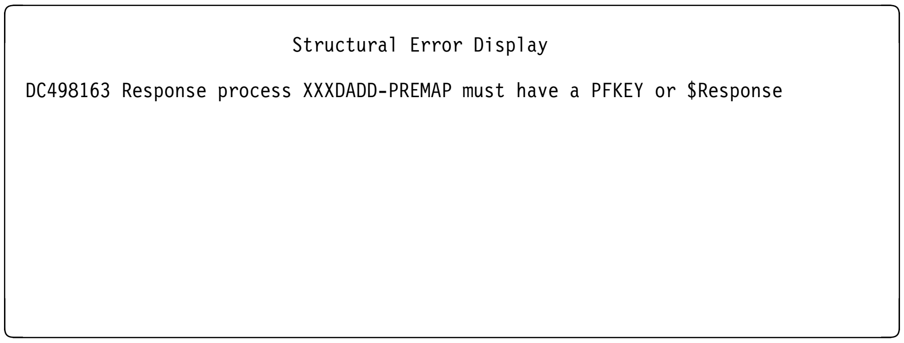
Some errors occur from the definition of the process module in ADSC. These are called **structural errors**.

When a message indicates that there are structural errors in your definition, access the **Structural Error Display** screen using the **Display messages** option from the **Compile** activity on the action bar.



```
   Add  Modify  Compile  Delete  Display  Switch
.______________________________________________________________________________.

              2  1. Compile           log Compiler
                 2. Display messages
             -------------------------  nternational, Inc.
              F3=Exit
             _________________________

   Dialog name . . . . . . .   XXXDADD
   Dialog version  . . . . .      1
   Dictionary name . . . . .   DEMO
   Dictionary node . . . . .   ________

   Screen  . . . . . . . . .   1  1. General options
                                  2. Assign maps
                                  3. Assign database
                                  4. Assign records and tables
                                  5. Assign process modules

Select 2, then Enter to see structural errors.

Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

Structural Error Display screen



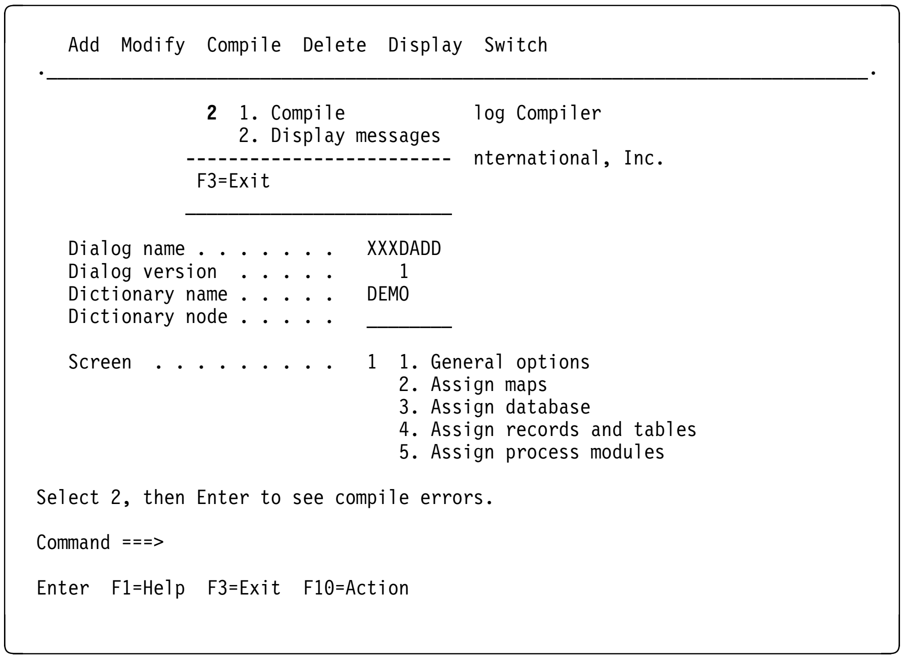
```
                         Structural Error Display

DC498163 Response process XXXDADD-PREMAP must have a PFKEY or $Response
```

Use this screen to find discrepancies between the process module and other components of the dialog. A missing or incorrect dialog component (for example, a subschema, map, or work record) causes compile-time errors for process commands that reference the component.

12.4.6.2 Display diagnostic messages

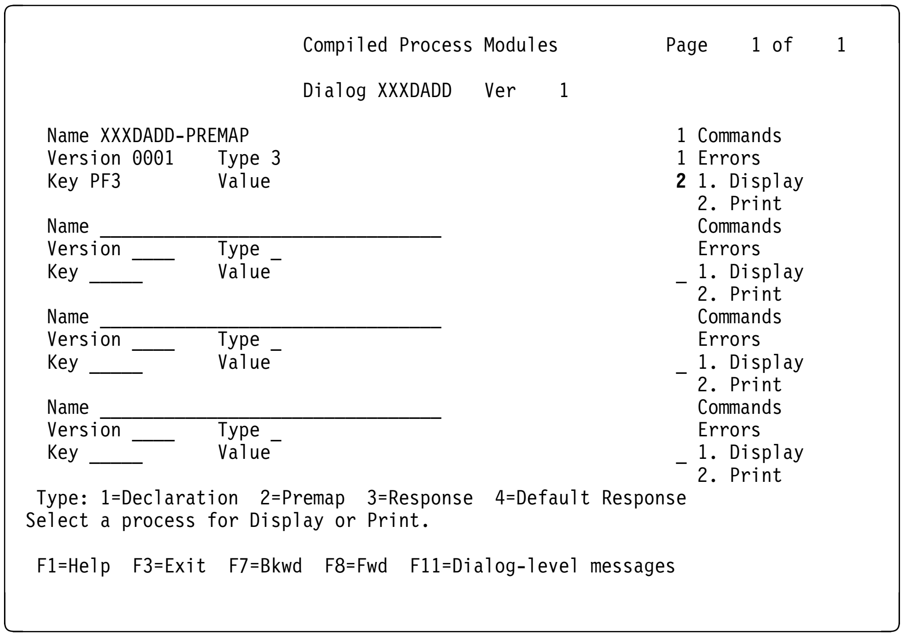
You display the ADSC **Dialog Process Source** screen to view a listing of the process module with diagnostic messages. You use this screen to find syntax errors in process module commands. Syntax errors (for example, mistyped commands and omitted periods) are the most frequent cause of compile-time errors.

Accessing the Dialog Process Source screen: To access the Dialog Process Source screen, When a message indicates that there are compile errors in your process code, use the **Display messages** option from the **Compile** activity on the action bar first to access the Compiled Process Modules screen.



```
   Add  Modify  Compile  Delete  Display  Switch
._____________________________________________________________________________.

               2  1. Compile            log Compiler
                  2. Display messages
             -------------------------  nternational, Inc.
              F3=Exit
             _________________________

   Dialog name . . . . . . .   XXXDADD
   Dialog version  . . . . .      1
   Dictionary name . . . . .   DEMO
   Dictionary node . . . . .   ________

   Screen  . . . . . . . . .   1  1. General options
                                  2. Assign maps
                                  3. Assign database
                                  4. Assign records and tables
                                  5. Assign process modules

Select 2, then Enter to see compile errors.

Command ===>

Enter  F1=Help  F3=Exit  F10=Action
```

Compiled Process Modules screen



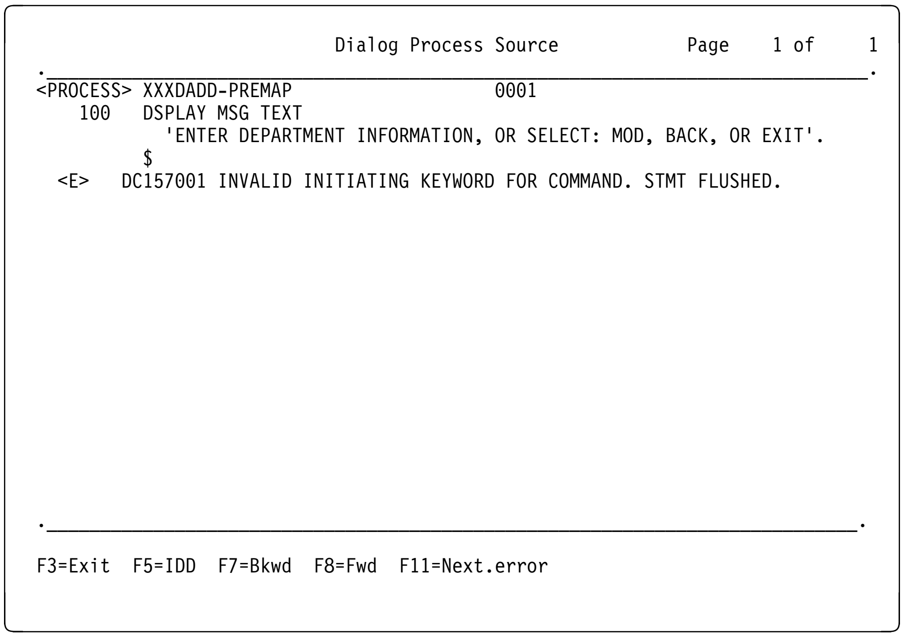
```
                         Compiled Process Modules         Page    1 of    1

                         Dialog XXXDADD   Ver    1

 Name XXXDADD-PREMAP                                       1 Commands
 Version 0001    Type 3                                    1 Errors
 Key PF3         Value                                     2 1. Display
                                                             2. Print
 Name ________________________________                       Commands
 Version ____    Type _                                      Errors
 Key _____       Value                                     _ 1. Display
                                                             2. Print
 Name ________________________________                       Commands
 Version ____    Type _                                      Errors
 Key _____       Value                                     _ 1. Display
                                                             2. Print
 Name ________________________________                       Commands
 Version ____    Type _                                      Errors
 Key _____       Value                                     _ 1. Display
                                                             2. Print
Type: 1=Declaration  2=Premap  3=Response  4=Default Response
Select a process for Display or Print.

F1=Help  F3=Exit  F7=Bkwd  F8=Fwd  F11=Dialog-level messages
```

The Compiled Process Modules screen shows the total number of commands in the process module and the number of errors encountered. You have the option of displaying the process code with the errors noted or printing them.

To display the process code, select **2** next to the process module you want to see and press [Enter].

The Dialog Process Source screen is then displayed.

Dialog Process Source screen



```
                            Dialog Process Source           Page    1 of     1
._____________________________________________________________________________.
<PROCESS> XXXDADD-PREMAP                       0001
    100   DSPLAY MSG TEXT
            'ENTER DEPARTMENT INFORMATION, OR SELECT: MOD, BACK, OR EXIT'.
          $
  <E>   DC157001 INVALID INITIATING KEYWORD FOR COMMAND. STMT FLUSHED.













.____________________________________________________________________________.

F3=Exit  F5=IDD  F7=Bkwd  F8=Fwd  F11=Next.error
```

Some errors cause error messages to be displayed for subsequent correct commands. For example, if you forget to put a period after a command, the next command line is incorrectly treated as a continuation of the first command.

Determining the causes for compile-time errors: When you attempt to add this process module to a dialog, compile errors will occur. You use the Dialog Process Source screen to determine the causes for compile-time errors. To do this, you:

1. **View the Compiler Process Modules screen** by selecting the **Display messages** option from the **Display** activity on the action bar of the Main Menu.

2. **View the Dialog Process Source screen** by selecting the **Display** activity from the Compiled Process Modules screen.

   Messages are displayed on this screen after statements that ADSC cannot compile. For example, assume that you made the following mistakes when you defined process module *XXX*DADD-RESPONSE earlier in this chapter:

```
READY USAGE MODE UPDATE.                    ◄-- There should be a dash (-)
IF AGR-CURRENT-RESPONSE NE SPACES               between USAGE and MODE.
AND FIELD DEPT-ID-0410 NOT CHANGED
THEN
    EXECUTE NEXT FUNCTION.

OBTAIN CALC DEPARTMENT                      ◄-- There should be a period at
IF DB-REC-NOT-FOUND                             the end of this statement.
THEN DO.

    STORE DEPARTMENT.
    DISPLAY MSG TEXT
        'DEPARTMENT ADDED'.
END.

DISPLAY MSG TEXT
    'TRY AGAIN, OR SELECT: MOD, BACK, OR EXIT'.
```

- A dollar sign ($) is displayed below the first character in a statement in error.
- The dash missing from the preceding USAGE-MODE keyword causes an error message.
- The period missing from the OBTAIN command causes the subsequence IF statement to be in error.
- The error in the above IF statement causes the error for the END statement.

In this example, the error in the USAGE-MODE keyword results in diagnostic messages for two correct commands that follow the OBTAIN command. You do not need to change these two correct commands.

3. **Page the screen back and forth** if necessary:
   - **Press** [PF8] to page forward.
   - **Press** [PF7] to page backward.
4. **Note what requires correction** in the process module source or in other dialog components.

   It is a good idea to **write down the errors** that you note.

When you have looked at your errors, press [PF5] to go directly to IDD. The process module is question will be displayed on the screen.

12.4.6.3 Correct structural errors

At compile time, errors in a process module can arise because of discrepancies between the process module and other dialog components. These are called **structural errors**.

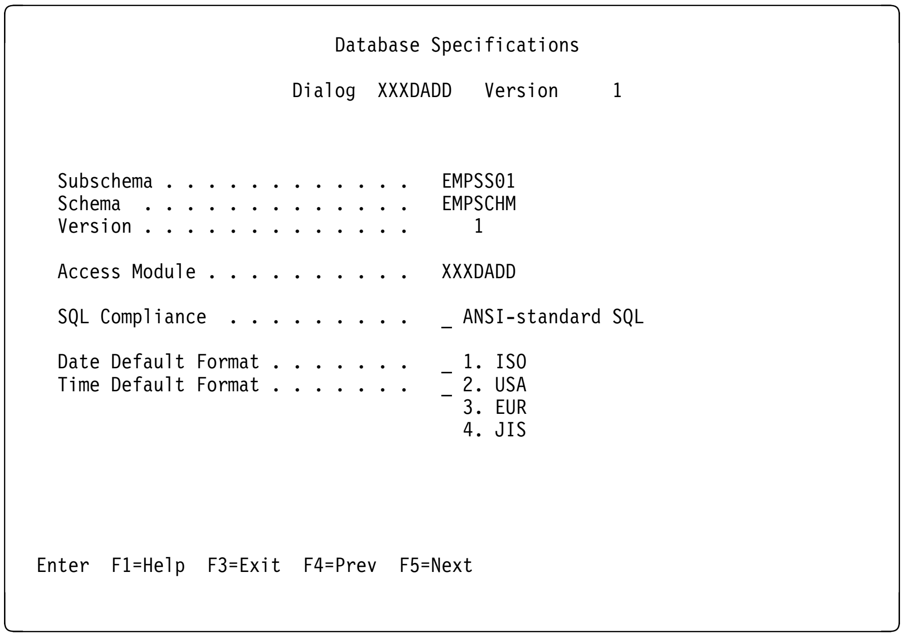
For example, you may have added the wrong subschema to the dialog. In this case, error messages are returned for process commands that reference records that exist in the correct subschema but don't exist in the specified subschema.

To correct discrepancies caused by other dialog components, you display the ADSC screen that associates the component with the dialog:

- For a **map**, display the Map Specifications screen.
- For a **subschema**, display the Database Specifications screen.
- For a **work record**, display the Records and Tables screen.

  Dialog *XXX*DADD doesn't contain a work record, so you don't need to check for work-record errors in the dialog.

Checking the subschema definition: For example, you check the subschema definition for dialog *XXX*DADD by displaying the Database Specifications screen:



```
                              Database Specifications

                          Dialog  XXXDADD   Version     1



   Subschema . . . . . . . . . . . .   EMPSS01
   Schema  . . . . . . . . . . . . .   EMPSCHM
   Version . . . . . . . . . . . . .      1

   Access Module . . . . . . . . . .   XXXDADD

   SQL Compliance  . . . . . . . . .   _ ANSI-standard SQL

   Date Default Format . . . . . . .   _ 1. ISO
   Time Default Format . . . . . . .   _ 2. USA
                                         3. EUR
                                         4. JIS




 Enter  F1=Help  F3=Exit  F4=Prev  F5=Next
```

When checking a component specification, you need to verify that both the correct **name** and **version number** are specified.

For example, dialog *XXX*DADD uses subschema EMPSS01. This subschema is defined in version 1 of schema EMPSCHM. Some sites define different copies of schema EMPSCHM. In this case, you check the Database Specifications screen:

Expected specification:

```
SUBSCHEMA.: EMPSS01      SCHEMA: EMPSCHM    VERSION:    1
```

Incorrect specification:

```
SUBSCHEMA.: EMPSS01      SCHEMA: EMPSCHM    VERSION:    2
```

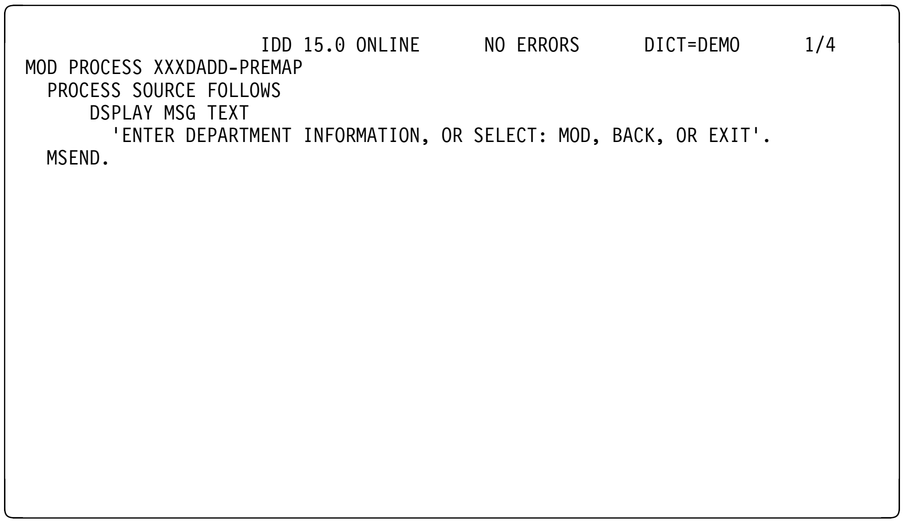
To correct any specifications, type the correct information over the previous specification. When dialog components are all specified correctly, you can proceed to correct any syntax errors in the process module.

12.4.6.4 Correct syntax errors

IDD is used to define process modules and to correct syntax errors in the process modules. To transfer from ADSC to IDD, you press [PF5] from the Dialog Process Source screen.

Note: When you transfer to IDD, your current ADSC session is saved. You can transfer back to ADSC and resume the saved definition session when you are done with IDD.

The IDD screen will display the process module in question.



```
                      IDD 15.0 ONLINE      NO ERRORS      DICT=DEMO      1/4
MOD PROCESS XXXDADD-PREMAP
  PROCESS SOURCE FOLLOWS
      DSPLAY MSG TEXT
        'ENTER DEPARTMENT INFORMATION, OR SELECT: MOD, BACK, OR EXIT'.
  MSEND.
```

Correct the errors in the process module and press [Enter] to store the revised code.

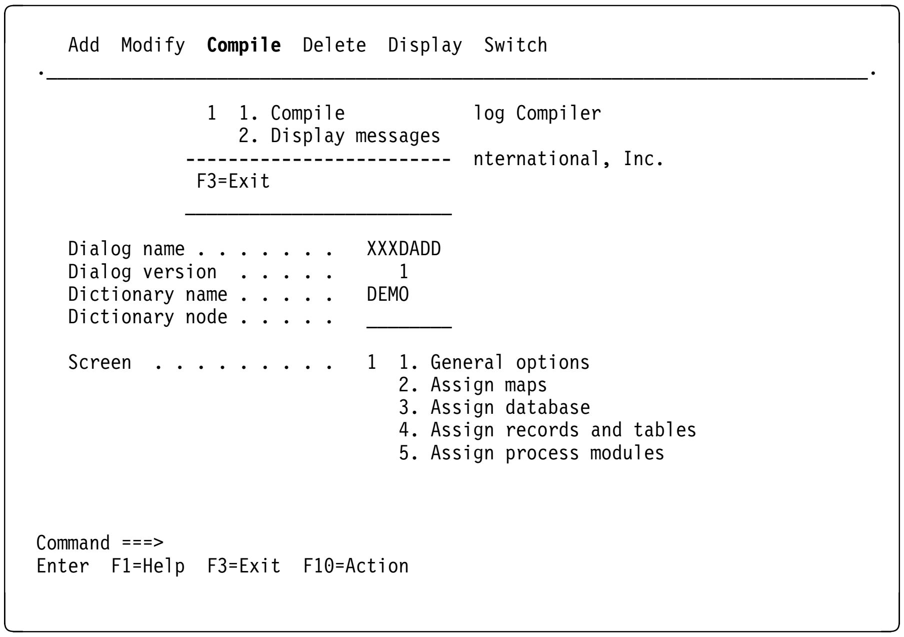
After you correct all errors in a process module, you can return to the dialog definition in ADSC. To do this, enter **end** command area at the top of the IDD screen:

```
-> end

                                                                  [Enter]
```

12.4.6.5 Update dialogs that use the process module

After you correct errors in a process module, you use ADSC to update the corrected process module in the dialog. When you do this, ADSC attempts to compile the source commands in the corrected process module. If the module compiles without errors, ADSC adds the process module to the dialog.

Recompile the dialog: One way to update modified process modules in a dialog is to recompile the dialog:



```
   Add  Modify  Compile  Delete  Display  Switch
.______________________________________________________________________________.

               1  1. Compile           log Compiler
                  2. Display messages
             -------------------------  nternational, Inc.
              F3=Exit
             _________________________

   Dialog name . . . . . . .   XXXDADD
   Dialog version  . . . . .      1
   Dictionary name . . . . .   DEMO
   Dictionary node . . . . .   ________

   Screen  . . . . . . . . .   1  1. General options
                                  2. Assign maps
                                  3. Assign database
                                  4. Assign records and tables
                                  5. Assign process modules



Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

When you press [Enter], ADSC attempts to compile corrected process modules. If there are no errors:

1. ADSC adds the compiled module to the dialog.
2. ADSC recompiles the dialog load module.
3. ADSC displays the Main Menu screen with a confirming message

When the process module is successfully corrected and updated to the dialog, you can resume definition and testing procedures for the dialog.

12.5 Executing the application

To test process logic in dialog *XXX*DADD, you will:

1. Invoke the Department application.
2. Display function ADDDEP, which executes dialog *XXX*DADD.
3. Try out process logic to add sample department records.

To invoke the Department application from DC/UCF, you enter the task code (*XXX*DEPT) for the application. For example, when using CA-IDMS/DC, you invoke the Department application as shown:

```
ENTER NEXT TASK CODE:
xxxdept

                                                        [Enter]
```

For more information on invoking the Department application, see 9.4, "Instructions for executing the application" on page 9-17.

While testing dialog *XXX*DADD in this chapter, it is recommended that you **write down the ID number** of each sample department record that you add to the database. This will help you later in this manual, when you need to locate, modify, and delete your sample department records.

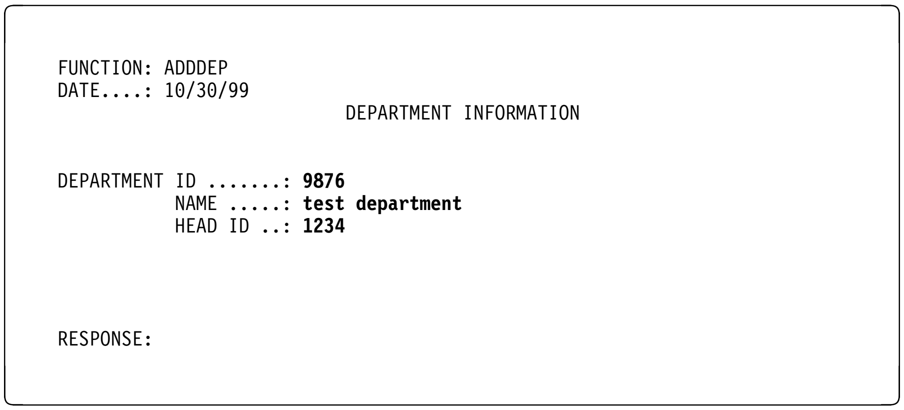
From DEPTMENU, you can display the ADDDEP function and try out dialog *XXX*DADD:

```
 x ADD

                                                        [Enter]
```

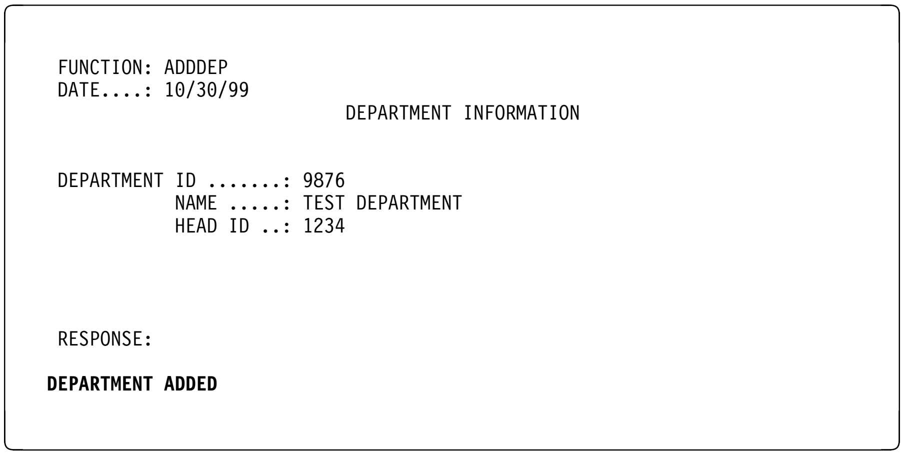
Enter sample information.

Note: You must specify a unique ID number for each department. Try entering the last four digits of your home phone number.

Entering sample information


```
FUNCTION: ADDDEP
DATE....: 10/30/99
                           DEPARTMENT INFORMATION


DEPARTMENT ID .......: 9876
           NAME .....: test department
           HEAD ID ..: 1234




RESPONSE:
```

Data is redisplayed: The data is redisplayed after it is added.


```
FUNCTION: ADDDEP
DATE....: 10/30/99
                           DEPARTMENT INFORMATION


DEPARTMENT ID .......: 9876
           NAME .....: TEST DEPARTMENT
           HEAD ID ..: 1234




RESPONSE:

DEPARTMENT ADDED
```

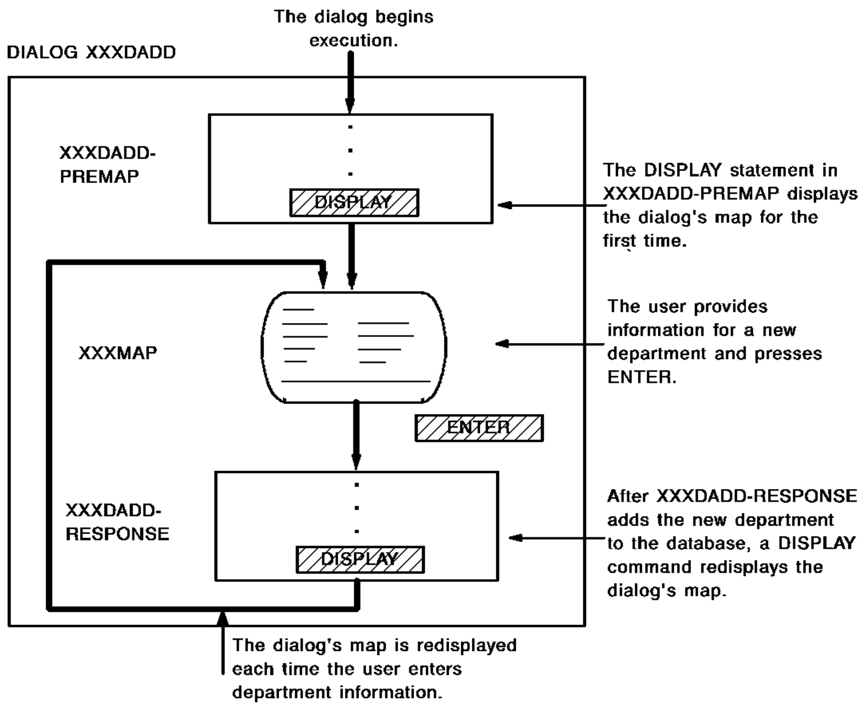
Notice the message you defined in response process *XXX*DADD-RESPONSE. It confirms that the redisplayed data has been added to the database.

When you press [Enter] to add a new department, response process *XXX*DADD-RESPONSE is executed. Statements in this response process test your input data and, for a new department, add the data to the sample database. The following diagram shows how components of dialog *XXX*DADD are executed at runtime when you use the dialog to add a department to the database. Dialog XXXDADD allows you to add new departments to the database. Execution remains in the dialog until the user requests another function (for example, MOD).

In dialog *XXX*DADD, you add a new department to the database by pressing [Enter]. To verify that your process logic keeps you from adding duplicate departments, try pressing [Enter] again to add the redisplayed department information.

After you press [Enter], the screen redisplays the department values along with an error message:

```
TRY AGAIN, OR SELECT: MOD, BACK, OR EXIT
```

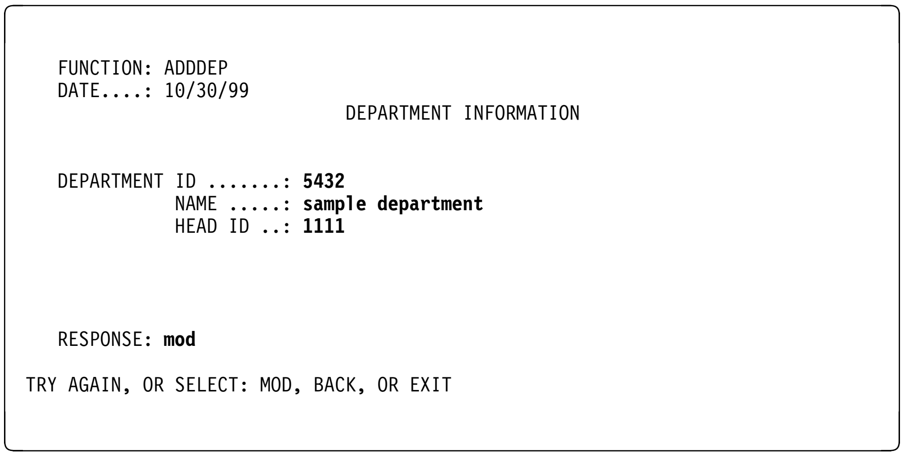

The dialog begins execution.
DIALOG XXXDADD
XXXDADD-PREMAP
DISPLAY
The DISPLAY statement in XXXDADD-PREMAP displays the dialog's map for the first time.
XXXMAP
The user provides information for a new department and presses ENTER.
ENTER
XXXDADD-RESPONSE
DISPLAY
After XXXDADD-RESPONSE adds the new department to the database, a DISPLAY command redisplays the dialog's map.
The dialog's map is redisplayed each time the user enters department information.

You defined this error message in process module *XXX*DADD-RESPONSE. The message is displayed whenever the specified department record already exists in the database. In this case, the duplicate record is not added to the database.

Try seeing what happens if you enter a new department record and also specify a valid response:

Continuing to test the XXXDADD dialog

- Type new values over the redisplayed department information.
- Enter a valid response name (for example, MOD).



```
   FUNCTION: ADDDEP
   DATE....: 10/30/99
                              DEPARTMENT INFORMATION


   DEPARTMENT ID .......: 5432
              NAME .....: sample department
              HEAD ID ..: 1111




   RESPONSE: mod

TRY AGAIN, OR SELECT: MOD, BACK, OR EXIT
```

Notice the message displayed when you added TEST DEPARTMENT above does not affect your current add operation.

After you press [Enter], dialog *XXX*DADD's screen is redisplayed with the following message:

```
DEPARTMENT ADDED
```

Mode of execution: Even when you specify a valid response, control remains in dialog *XXX*DADD when you add a new department. This type of execution is called **STEP mode** execution. In STEP mode, the screen is always redisplayed with a confirming message after a successful add, modify, or delete operation.

Dialog *XXX*DADD executes in STEP mode because of the following conditional command that you coded at the beginning of response process *XXX*DADD-RESPONSE:

```
IF AGR-CURRENT-RESPONSE NE SPACES   ◄-- If the user specifies
                                        a valid response
AND DEPT-ID-0415 NOT CHANGED        ◄-- and doesn't enter a new
                                        department ID,
THEN
    EXECUTE NEXT FUNCTION.          ◄-- control transfers to the
                                        next function.
```

Dialogs do not have to execute in STEP mode. As an application developer, you can make use of the following execution strategies when developing an application:

- Dialog response processes can execute in **FAST mode**. In this case, you write process logic that can handle necessary data operations and then transfer control without first redisplaying the screen. If the user inputs errors, the current screen can be redisplayed with an error message.

  This strategy benefits experienced users, who can input data and advance directly to the next function without any extra keystrokes.

- Application functions can be **immediately executable**. In this case, the runtime system executes the function as soon as the user requests transfer to the function. Process logic is not executed before the transfer occurs.

  This strategy ensures that users do not inadvertently alter the database when transferring *to* another function. To make a function immediately executable, you use the ADSA Response Definition screen.

  >> For more information on immediately executable functions, see the *CA-ADS Reference*.

Another test: Now, test whether your process logic allows users to transfer control to other functions. According to your application design, for example, you can transfer to function MODDEP in either of the following ways:

- **You can press the control key** ([PF5]) associated with MODDEP.

- **You can enter the associated response name** (MOD) in the map's response field and press [Enter].

When you request transfer to MODDEP:

1. The runtime system stores the name (MOD) of the specified response in system-supplied element AGR-CURRENT-RESPONSE.
2. The runtime system executes process module *XXX*DADD-RESPONSE for the dialog, whether you pressed [Enter] or [PF5] to request transfer.

If you haven't already, transfer to MODDEP:

```
RESPONSE: mod

                                                               [Enter]
```

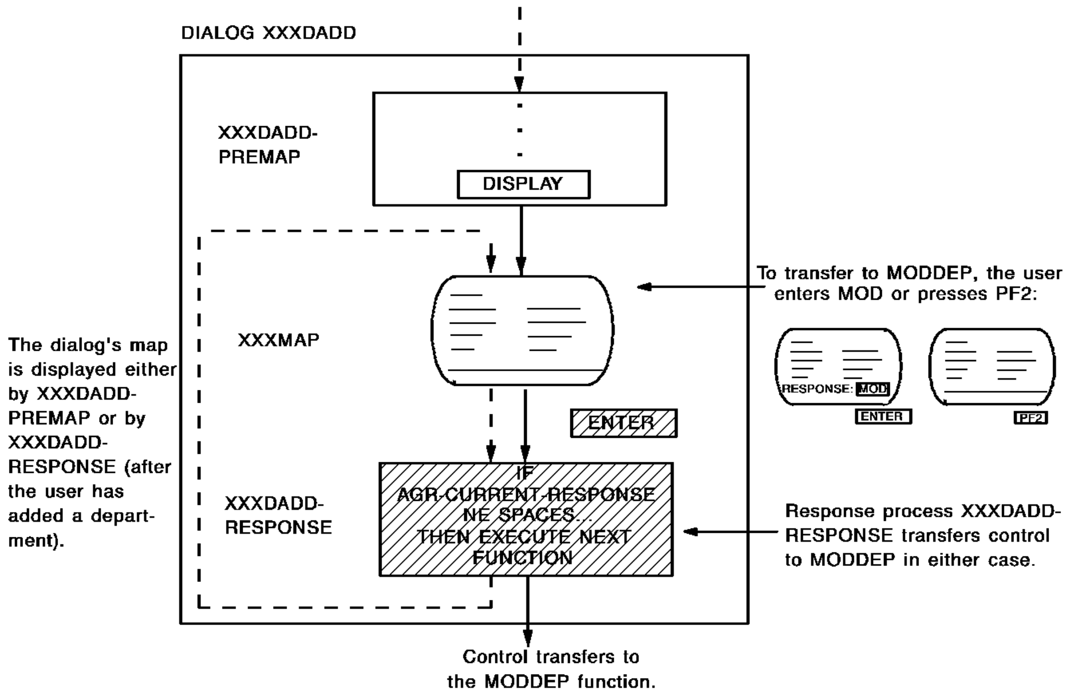
Since you didn't enter a department record along with the response name, control transferred to the indicated function. This processing was handled by the following conditional command at the beginning of process module *XXX*DADD-RESPONSE:

```
IF AGR-CURRENT-RESPONSE NE SPACES   ◄-- You entered a valid response
AND DEPT-ID-0415 NOT CHANGED        ◄-- and didn't enter a new department;
THEN
    EXECUTE NEXT FUNCTION.          ◄-- therefore, control transferred.
```

How components are executed: The following diagram shows how components for dialog *XXX*DADD are executed at runtime when you request transfer to another application function. Dialog XXXDADD has a response process associated with [Enter]. No response process is associated with [PF5], the control key defined for the MOD response.

It is unnecessary to test function MODDEP at this time since you have not yet defined any process logic for the associated dialog. You can exit from the Department application by specifying the EXIT response in the RESPONSE field:

```
 RESPONSE:  exit

                                                               [Enter]
```


DIALOG XXXDADD
XXXDADD-PREMAP
DISPLAY
XXXMAP
XXXDADD-RESPONSE
ENTER
IF AGR-CURRENT-RESPONSE NE SPACES... THEN EXECUTE NEXT FUNCTION
The dialog's map is displayed either by XXXDADD-PREMAP or by XXXDADD-RESPONSE (after the user has added a department).
To transfer to MODDEP, the user enters MOD or presses PF2:
RESPONSE: MOD
ENTER
PF2
Response process XXXDADD-RESPONSE transfers control to MODDEP in either case.
Control transfers to the MODDEP function.

12.6 Summary

In this chapter, you enhanced dialog *XXX*DADD so that it performs all operations necessary for final production use. You added processing logic to the dialog by performing the following steps:

1. **You defined two process modules** by using the IDD menu facility:
   - Process module *XXX***DADD-PREMAP** displays the dialog's map with a message for the user.
   - Process module *XXX***DADD-RESPONSE** evaluates end-user input and stores department information in the database.
2. **You added the process modules to dialog** ***XXX*****DADD** by using ADSC:
   - As a **premap process**, process module *XXX*DADD-PREMAP will be executed whenever dialog *XXX*DADD begins execution.
   - As a **response process** associated with [Enter],R process module *XXX*DADD-RESPONSE will be executed whenever the user presses [Enter] input data on the map for dialog *XXX*DADD.
3. **You executed the Department application** to see how the process modules affect execution of dialog *XXX*DADD:
   - **You added new departments**, one at a time, by entering department information on the dialog's map.
   - **You transferred to function MODDEP** when you were finished using ADDDEP to add new departments.

Both developers and users can execute a dialog to test its process logic. Based on tests, developers and users often suggest modifications to process logic.

For example, after testing a fully developed dialog, users might find it confusing that the current screen is redisplayed if it contains input errors when the user enters BACK or EXIT. As an application developer, you might modify the application so that BACK and EXIT exit the user unconditionally from a dialog. To do this, you would use ADSA to modify BACK and EXIT, making the associated functions immediately executable.

As another example, users testing dialog *XXX*DADD might request that, after a new department is successfully added, the screen be redisplayed without the department's information. This modification would make it more obvious that the redisplayed screen can be used to add another department.

In the next chapter, you will modify process module *XXX*DADD-RESPONSE so that it redisplays an initialized screen after a new department is added to the database.

Chapter 13. Modifying Process Logic in a Dialog

13.1 Introduction

In the previous chapter, you defined a premap and a response process for dialog *XXX*DADD. You then executed the Department application to see how these processes affect dialog *XXX*DADD at runtime.

Based on runtime tests, it may be necessary to modify a dialog's premap or response process. This chapter provides instructions for modifying process module *XXX*DADD-RESPONSE, which you defined in the previous chapter. This chapter includes:

- An overview of modifying process logic
- Steps for modifying process modules
- Steps for updating modified process modules in dialogs
- A summary of what you've accomplished in this chapter

13.2 Overview

You can modify a dialog's premap or response process at any time in an application's life cycle. For example, assume that users who test out dialog *XXX*DADD ask you to change processing so that successfully added department data is not redisplayed to the user. To accomplish this change, you need to modify process module *XXX*DADD-RESPONSE, which evaluates, stores, and redisplays department data input by users.

Initializing the map: To initialize map *XXX*MAP after new department data is added to the database, you will add an INITIALIZE command to process *XXX*DADD-RESPONSE. The modified process module is shown below.

```
READY USAGE-MODE UPDATE.
IF AGR-CURRENT-RESPONSE NE SPACES
AND FIELD DEPT-ID-0410 NOT CHANGED
THEN
    EXECUTE NEXT FUNCTION.

OBTAIN CALC DEPARTMENT.
IF DB-REC-NOT-FOUND
THEN DO.

    STORE DEPARTMENT.
    INITIALIZE (DEPARTMENT).  ◄--- This command initializes the
    DISPLAY MSG TEXT               DEPARTMENT record buffer after the values
        'DEPARTMENT ADDED'.        in the buffer are stored in the database.
END.

DISPLAY MSG TEXT
    'TRY AGAIN, OR SELECT: MOD, BACK, OR EXIT'.
```

When you modify process module *XXX*DADD-RESPONSE in this chapter, you will:

1. **Modify source commands for the process module** by using the IDD menu facility. You used the IDD menu facility in Chapter 12, “Adding Process Logic to a Dialog” on page 12-1, to define process module *XXX*DADD-RESPONSE.
2. **Update the modified process module in dialogs that use the process** by using the CA-ADS dialog compiler (ADSC). In this chapter, you will use ADSC to update modified process module *XXX*DADD-RESPONSE in dialog *XXX*DADD.

Steps for modifying a process module and updating the modified process module in a dialog are presented below.

13.3 Modifying process modules using IDD

You can use IDD menu facility screens to modify a process module. In this chapter, you will modify process module *XXX*DADD-RESPONSE by performing the following steps:

1. Retrieve the process module definition.
2. Modify process module source statements.

These steps are discussed below.

13.3.1 Step 1: Retrieve the process module definition

In order to retrieve a process module, you must first invoke the IDD menu facility by using its task code (for example, IDDMT). For example, when using CA-IDMS/DC, you invoke the IDD menu facility as shown:

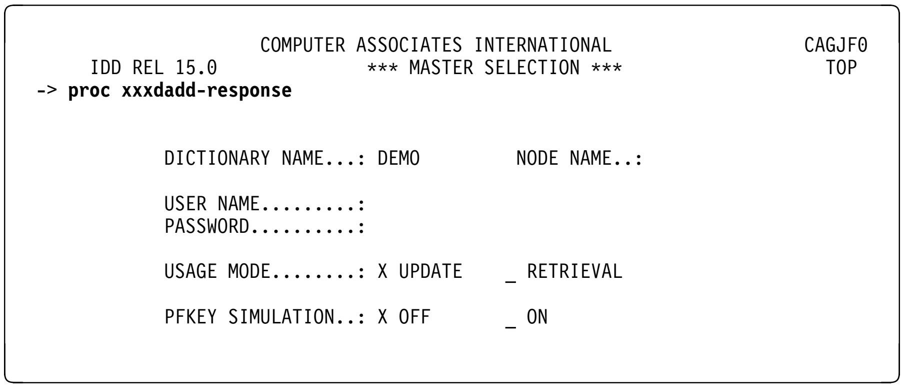
```
ENTER NEXT TASK CODE:
iddmt
                                                              [Enter]
```

After you invoke the IDD menu facility, you may need to provide signon information.

For more information on invoking and signing on to the IDD menu facility, see 12.3.1, “Step 1: Invoke the IDD menu facility” on page 12-10.

You display a process module definition on the **Process Entity** screen. You can transfer to this screen and request display of an existing process module at the same time. To do this, enter the identifier for the screen (PROC) in the command area of an IDD menu facility screen, followed by the name of the process module.

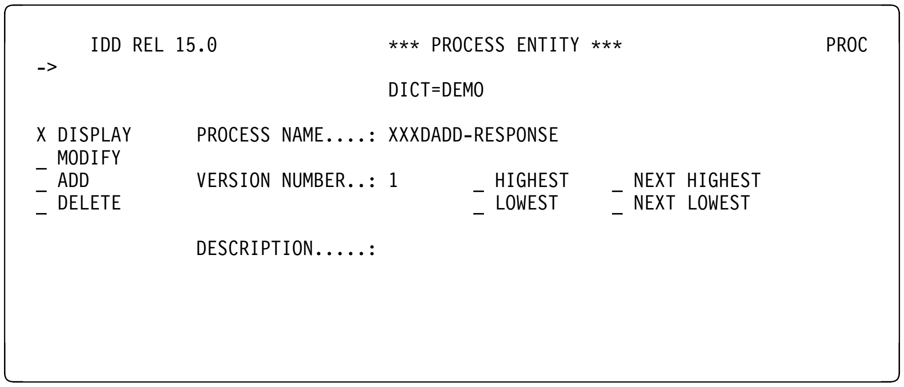
Displaying a process module definition: For example, you can display process module *XXX*DADD-RESPONSE from the Master Selection screen by specifying PROC and the process module name in the command area.



```
                 COMPUTER ASSOCIATES INTERNATIONAL                  CAGJF0
     IDD REL 15.0             *** MASTER SELECTION ***                TOP
 -> proc xxxdadd-response


           DICTIONARY NAME...: DEMO         NODE NAME..:

           USER NAME.........:
           PASSWORD..........:

           USAGE MODE........: X UPDATE    _ RETRIEVAL

           PFKEY SIMULATION..: X OFF       _ ON
```

The process module definition is displayed.

Process Entity screen



```
     IDD REL 15.0                *** PROCESS ENTITY ***                  PROC
->
                                 DICT=DEMO

X DISPLAY      PROCESS NAME....: XXXDADD-RESPONSE
_ MODIFY
_ ADD          VERSION NUMBER..: 1       _ HIGHEST    _ NEXT HIGHEST
_ DELETE                                 _ LOWEST     _ NEXT LOWEST

               DESCRIPTION.....:
```

After you press [Enter], IDD displays basic information about the specified process module on the Process Entity screen, along with a message like:

```
PROCESS 'XXXDADD-RESPONSE' VERSION 1 DISPLAYED
```

You can make modifications to specifications on the Process Entity screen, if necessary, and then press [Enter] to store the modified information.

13.3.2 Step 2: Modify source statements

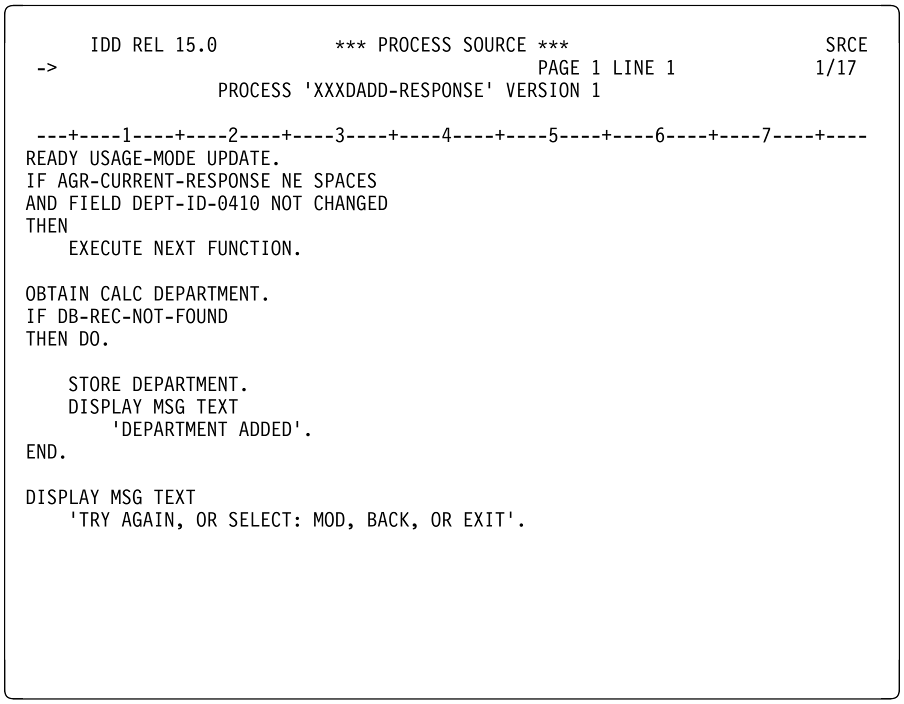
You defined process module *XXX*DADD-RESPONSE in Chapter 12, “Adding Process Logic to a Dialog” on page 12-1. In this step, you will modify source commands in the process module so that the dialog's screen is initialized for redisplay after the user inputs a new department record.

To modify source commands for a process module, you use the **Process Source** screen. You can display the Process Source screen for process module *XXX*DADD-RESPONSE as shown.

Displaying the Process Source screen: Type the screen identifier (SRCE) in the command area.

```
-> srce
                                                                    [Enter]
```

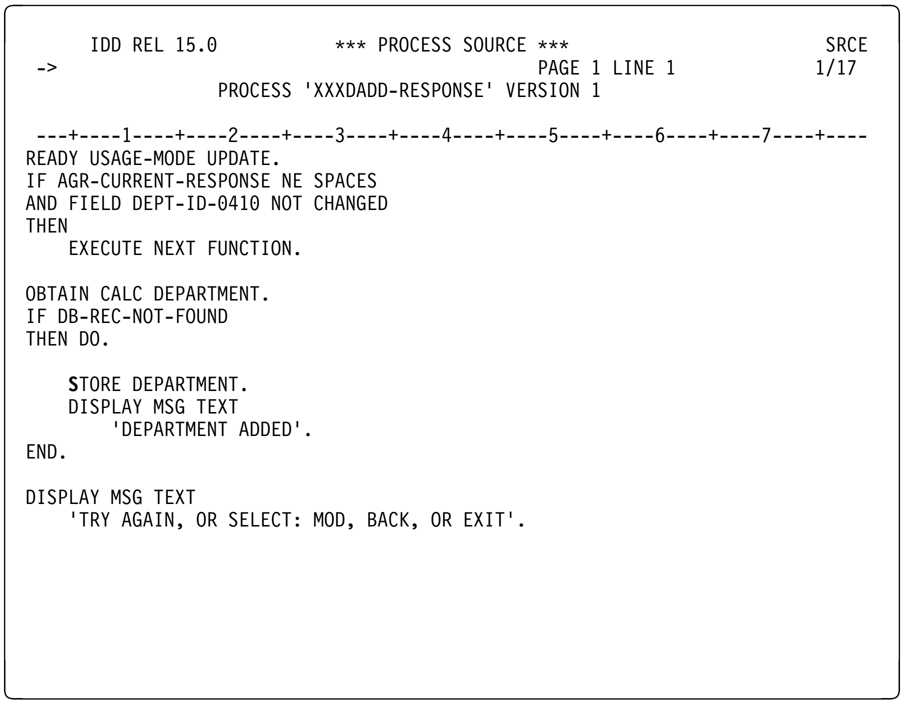
The Process Source screen is displayed.



```
      IDD REL 15.0           *** PROCESS SOURCE ***                        SRCE
 ->                                                PAGE 1 LINE 1            1/17
                  PROCESS 'XXXDADD-RESPONSE' VERSION 1

 ---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----
READY USAGE-MODE UPDATE.
IF AGR-CURRENT-RESPONSE NE SPACES
AND FIELD DEPT-ID-0410 NOT CHANGED
THEN
    EXECUTE NEXT FUNCTION.

OBTAIN CALC DEPARTMENT.
IF DB-REC-NOT-FOUND
THEN DO.

    STORE DEPARTMENT.
    DISPLAY MSG TEXT
        'DEPARTMENT ADDED'.
END.

DISPLAY MSG TEXT
    'TRY AGAIN, OR SELECT: MOD, BACK, OR EXIT'.
```

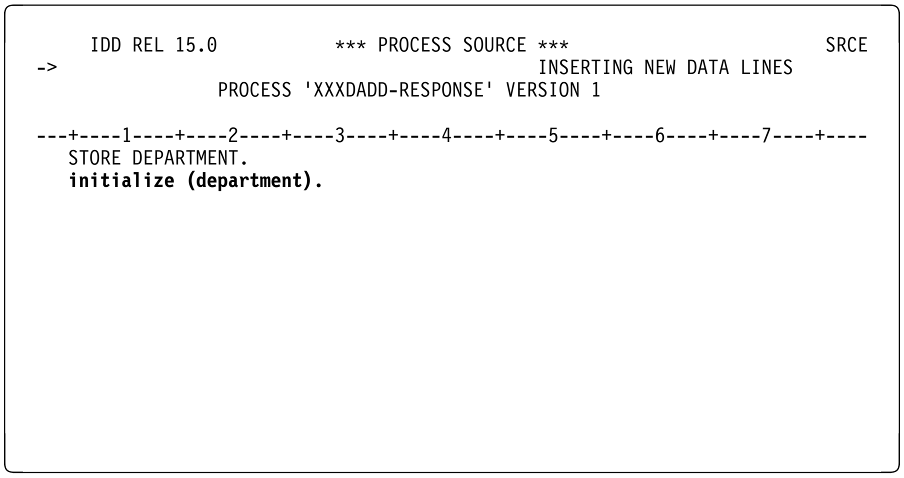
Modifying the process module: To modify process module *XXX*DADD-RESPONSE, you need to insert an INITIALIZE RECORDS command in the source commands for the process module:

1. Place the cursor on the line after which new statements are to be added.
2. Press the control key that inserts new lines on IDD screens (a different control key may be defined at your site).



```
      IDD REL 15.0           *** PROCESS SOURCE ***                        SRCE
 ->                                             PAGE 1 LINE 1             1/17
                PROCESS 'XXXDADD-RESPONSE' VERSION 1

 ---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----
READY USAGE-MODE UPDATE.
IF AGR-CURRENT-RESPONSE NE SPACES
AND FIELD DEPT-ID-0410 NOT CHANGED
THEN
    EXECUTE NEXT FUNCTION.

OBTAIN CALC DEPARTMENT.
IF DB-REC-NOT-FOUND
THEN DO.

    STORE DEPARTMENT.
    DISPLAY MSG TEXT
        'DEPARTMENT ADDED'.
END.

DISPLAY MSG TEXT
    'TRY AGAIN, OR SELECT: MOD, BACK, OR EXIT'.
```

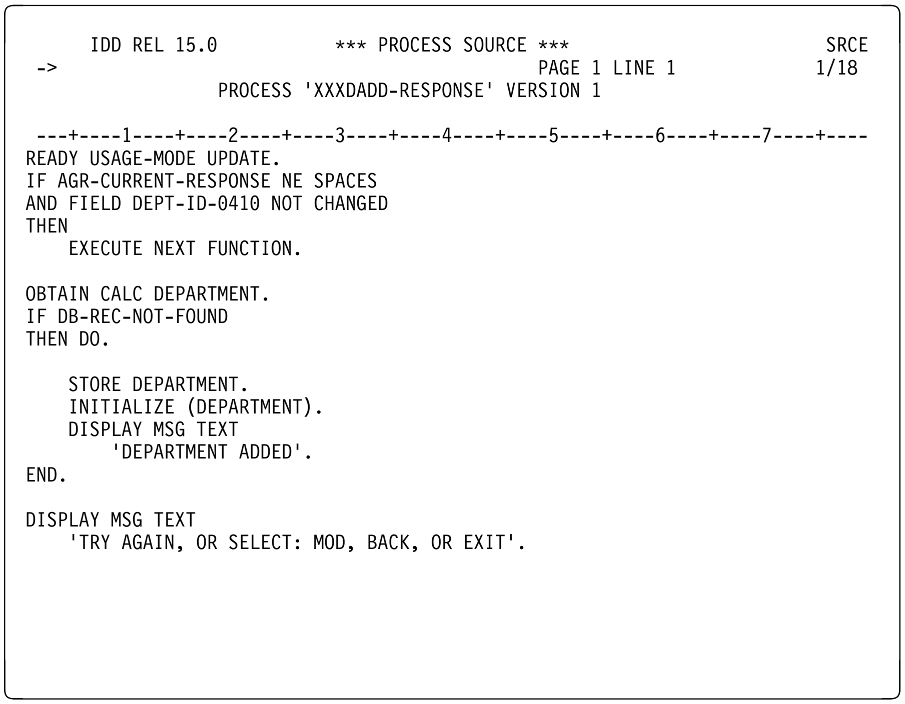
Add the INITIALIZE statement and press the control key that applies changed or inserted lines to IDD screens (a different key may be defined at your site).



```
      IDD REL 15.0           *** PROCESS SOURCE ***                        SRCE
 ->                                             INSERTING NEW DATA LINES
                PROCESS 'XXXDADD-RESPONSE' VERSION 1

 ---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----
    STORE DEPARTMENT.
    initialize (department).
```

The updated process statements are displayed.

Press [Enter] to store the updated process module in the data dictionary:



```
      IDD REL 15.0           *** PROCESS SOURCE ***                        SRCE
 ->                                            PAGE 1 LINE 1             1/18
                PROCESS 'XXXDADD-RESPONSE' VERSION 1

 ---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----
READY USAGE-MODE UPDATE.
IF AGR-CURRENT-RESPONSE NE SPACES
AND FIELD DEPT-ID-0410 NOT CHANGED
THEN
    EXECUTE NEXT FUNCTION.

OBTAIN CALC DEPARTMENT.
IF DB-REC-NOT-FOUND
THEN DO.

    STORE DEPARTMENT.
    INITIALIZE (DEPARTMENT).
    DISPLAY MSG TEXT
        'DEPARTMENT ADDED'.
END.

DISPLAY MSG TEXT
    'TRY AGAIN, OR SELECT: MOD, BACK, OR EXIT'.
```

When you press [Enter], IDD updates the process module in the data dictionary. Then, the Process Source screen displays a message like:

```
PROCESS 'XXXDADD-RESPONSE' VERSION 1 MODIFIED
```

>> For more information on inserting lines in a process module, see *CA-IDMS Online Compiler Text Editor*.

Exit from IDD: After you successfully modify process module *XXX*DADD-RESPONSE, you can exit from IDD. In this sample session, you will transfer directly from IDD to ADSC to update dialog *XXX*DADD. To do this, use the SWITCH command, followed by the task code for ADSC.

You enter the SWITCH command in the command area of any IDD menu facility screen:

```
-> switch adsct
                                                                  [Enter]
```

After you exit from IDD, you can use ADSC to update the modified process module in any dialog that uses it.

13.4 Updating modified process modules in dialogs using ADSC

In the above step, you used the IDD menu facility to modify process module *XXX*DADD-RESPONSE. Now, you will recompile dialog *XXX*DADD, which uses this process module. When you recompile the dialog, ADSC compiles the modified source statements for the process module.

You will perform the following steps:

1. Retrieve and check out the dialog to be updated.
2. Recompile the dialog.

After you recompile the dialog and exit from ADSC, you can execute the application to test the modified dialog.

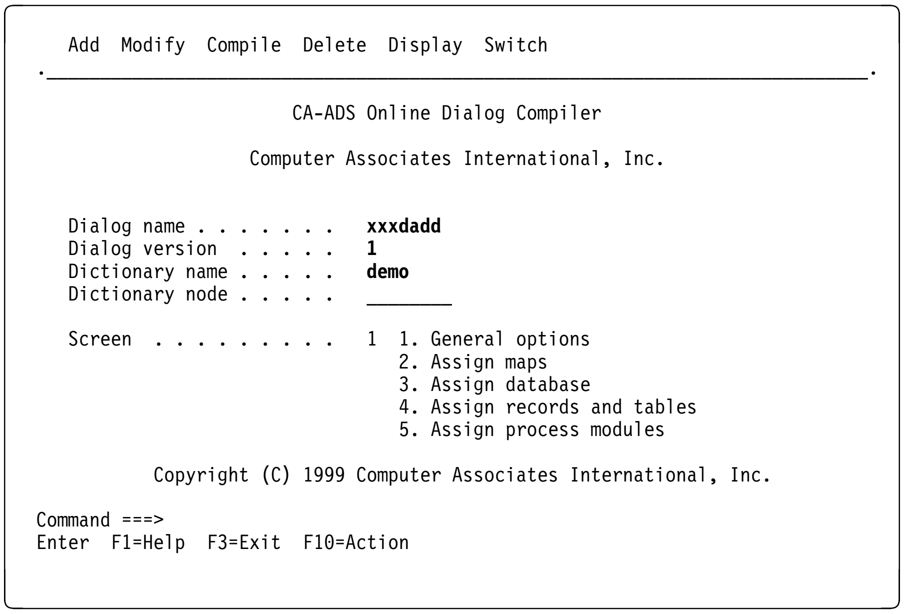
13.4.1 Step 1: Retrieve and check out the dialog

In order to retrieve a dialog definition, you must be using ADSC.

If you *did not* transfer directly to ADSC earlier in this chapter when you exited from IDD, you need to invoke ADSC by using the task code (for example, ADSCT) for ADSC. For more information on invoking ADSC, see 9.3.1, "Step 1: Invoke ADSC" on page 9-8.

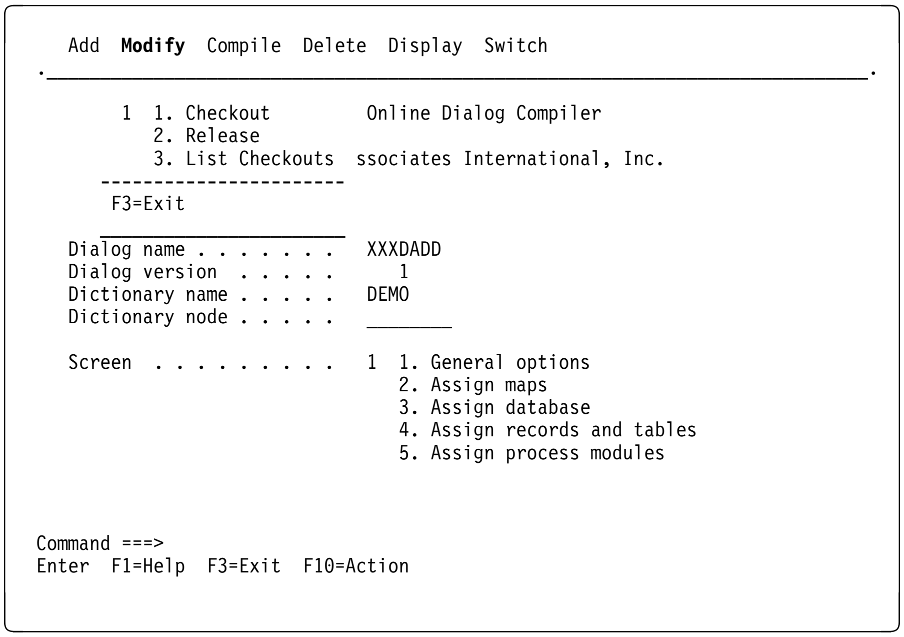
ADSC begins by displaying the **Main Menu** screen. You use the Main Menu screen to retrieve a dialog definition for update.

Retrieving the dialog: For example, you can retrieve dialog *XXX*DADD:


```
   Add  Modify  Compile  Delete  Display  Switch
._____________________________________________________________________________.

                      CA-ADS Online Dialog Compiler

                  Computer Associates International, Inc.


   Dialog name . . . . . . .   xxxdadd
   Dialog version  . . . . .   1
   Dictionary name . . . . .   demo
   Dictionary node . . . . .   ________

   Screen  . . . . . . . . .   1  1. General options
                                  2. Assign maps
                                  3. Assign database
                                  4. Assign records and tables
                                  5. Assign process modules

           Copyright (C) 1999 Computer Associates International, Inc.

Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

Checking out an application: To check the application out for modification, position the cursor on the **Modify** item on the action bar and press [Enter]. You can position the cursor on **Modify** by:

- Pressing [PF10] to move to the action bar and then tabbing to **Modify** and pressing [Enter]
- Tabbing to **Modify** and pressing [Enter]
- Typing **modify** on the command line and pressing [Enter]


```
   Add  Modify  Compile  Delete  Display  Switch
._____________________________________________________________________________.

        1  1. Checkout        Online Dialog Compiler
           2. Release
           3. List Checkouts  ssociates International, Inc.
      -----------------------
       F3=Exit
      _______________________
   Dialog name . . . . . . .   XXXDADD
   Dialog version  . . . . .      1
   Dictionary name . . . . .   DEMO
   Dictionary node . . . . .   ________

   Screen  . . . . . . . . .   1  1. General options
                                  2. Assign maps
                                  3. Assign database
                                  4. Assign records and tables
                                  5. Assign process modules



Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

Press [Enter] to check the dialog out.

Note: If the dialog has been checked out to another developer and has not been either compiled or released by that developer, you will not be able to check it out. The other developer must release the dialog.

After you press [Enter] to retrieve the dialog, ADSC displays basic information about dialog *XXX*DADD on the Main Menu screen along with the following message:

```
NO ERRORS DETECTED
```

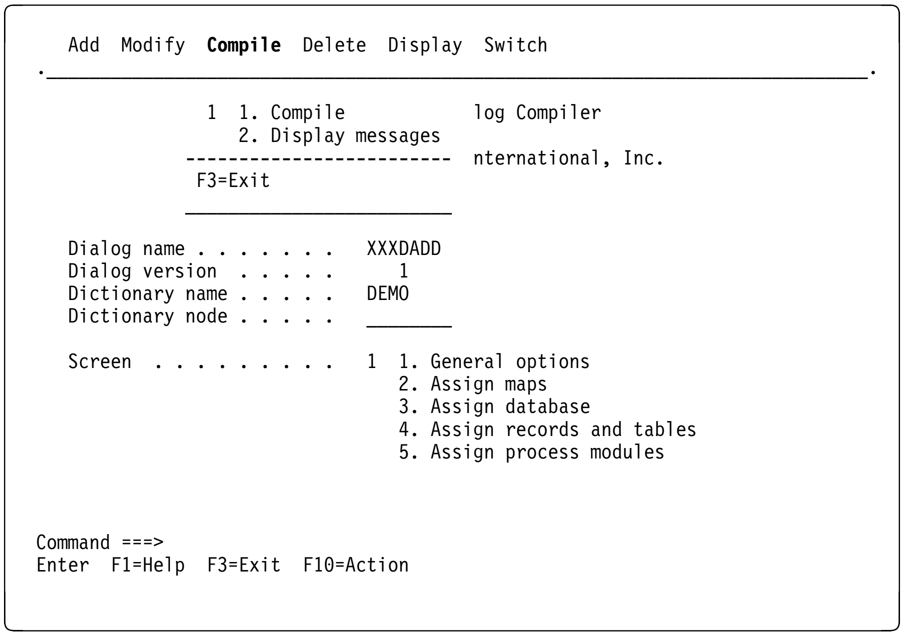
For more information on retrieving dialogs, see 11.10, “Step 1: Retrieve dialog XXXDADD” on page 11-22.

After you successfully check out *XXX*DADD, you can update the modified process module in the dialog.

13.4.2 Step 2: Recompile the dialog

Earlier in this chapter, you modified source statements for process module *XXX*DADD-RESPONSE. In order to include the modified process module in dialog *XXX*DADD, you must recompile the dialog. When you recompile the dialog, the modified source module is compiled. If the compilation is successful, the compiled process module is included in the dialog load module.

To recompile dialog *XXX*DADD, select the **Compile** activity from the action bar on the Main Menu of the dialog compiler.



```
   Add  Modify  Compile  Delete  Display  Switch
.______________________________________________________________________________.

             1  1. Compile           log Compiler
                2. Display messages
            -------------------------  nternational, Inc.
             F3=Exit
            _________________________

   Dialog name . . . . . . .   XXXDADD
   Dialog version  . . . . .      1
   Dictionary name . . . . .   DEMO
   Dictionary node . . . . .   ________

   Screen  . . . . . . . . .   1  1. General options
                                  2. Assign maps
                                  3. Assign database
                                  4. Assign records and tables
                                  5. Assign process modules



Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

After you press [Enter], ADSC displays either:

- A confirming message when process modules contain no errors and the dialog load module has been successfully recompiled.
- An error message when errors in a process module prevent ADSC from successfully recompiling the dialog.

  In this case, you correct errors in the process module as described in 12.4.6, “Correct errors in process modules” on page 12-34.

After you successfully recompile the dialog, you can exit from ADSC by pressing [PF3].

13.4.3 Execute the application

You can now execute the Department application to see how adding an INITIALIZE RECORDS command to process module *XXX*DADD-RESPONSE affects dialog *XXX*DADD at runtime.

You can invoke the Department application from DC/UCF by entering the task code (for example, *XXX*DEPT) for the application. For example, when using CA-IDMS/DC, you invoke the Department application as shown:

```
ENTER NEXT TASK CODE:
xxxdept

                                              [Enter]
```

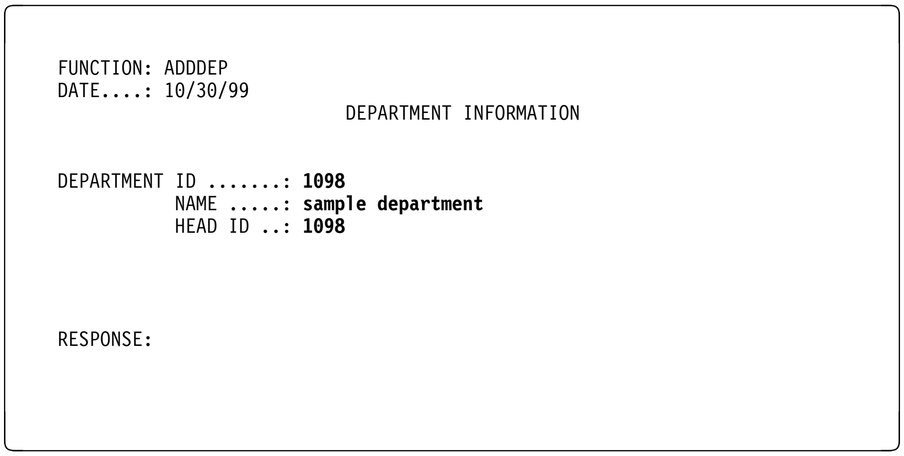
For more information on invoking the application, see 9.4, “Instructions for executing the application” on page 9-17.

According to your application design, ADDDEP is the only function that executes dialog *XXX*DADD.

The ADDDEP function allows you to add new department information. When you used the ADDDEP function to add sample departments in 12.5, “Executing the application” on page 12-42, your sample data was redisplayed to you after you added the data to the application database by pressing [Enter].

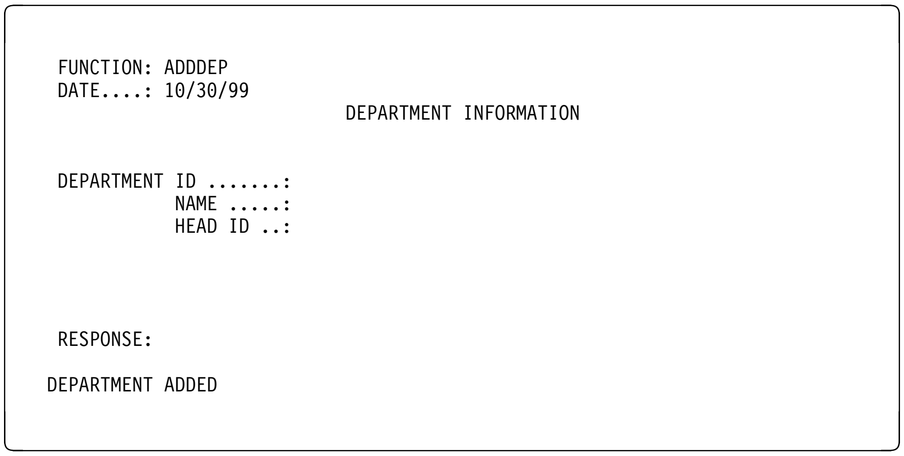
Entering sample information: Display ADDDEP and then try adding a sample department now that you have modified process module *XXX*DADD-RESPONSE.

Note: Enter different department data than shown here.



```
FUNCTION: ADDDEP
DATE....: 10/30/99
                           DEPARTMENT INFORMATION


DEPARTMENT ID .......: 1098
           NAME .....: sample department
           HEAD ID ..: 1098




RESPONSE:
```

Screen is redisplayed: An initialized screen is displayed after you add new department data.



```
 FUNCTION: ADDDEP
 DATE....: 10/30/99
                            DEPARTMENT INFORMATION


 DEPARTMENT ID .......:
            NAME .....:
            HEAD ID ..:




 RESPONSE:

DEPARTMENT ADDED
```

When you are finished testing the ADDDEP function, you can exit from the Department application by entering the EXIT response in the screen's RESPONSE field.

13.5 Summary

In this and previous chapters, you developed dialog *XXX*DADD to add new departments to the database for the Department application by performing the following steps:

1. **You defined dialog *XXX*DADD as a skeleton dialog** for the prototype Department application in Chapter 9, “Defining Dialogs Using ADSC” on page 9-1 by adding map *XXX*MAP to the dialog.
2. **You enhanced the dialog to full production capability** in Chapter 12, “Adding Process Logic to a Dialog” on page 12-1, by adding a subschema, premap process, and response process to the dialog.
3. **You completed the dialog** in this chapter by modifying the dialog's response process based on end-user testing of the dialog.

Now that you have completed dialog *XXX*DADD, you can proceed to enhance dialog *XXX*DUPD by defining a work record and modules of process code for the dialog.

Chapter 14. Defining Work Records Using IDD

14.1 Introduction

In previous chapters, you used the IDD menu facility to define and modify process modules for use in dialog *XXX*DADD. You also can use IDD to define work records for use by dialogs.

Instructions for defining work records are provided in this chapter. The sample work record defined in this chapter will be used by process modules that you will define for dialog *XXX*DUPD in Chapter 15, “Completing the Department Application” on page 15-1. This chapter includes:

- An overview of defining work records for CA-ADS applications
- Instructions for defining work records
- A summary of what you've accomplished in this chapter

Note: At some sites, all records and elements are defined by database administrators (DBAs). Even if this is the case at your site, go ahead and read this chapter to find out more about records in the CA-ADS environment.

14.2 Overview

Each item of data used and stored by your application must be defined as an **element** in the data dictionary. Most elements describe database values. For example, DEPT-ID-0410, DEPT-NAME-0410, and DEPT-HEAD-ID-0410 are elements that store information about a department in the database.

Elements are grouped together in **records**. For example, elements DEPT-ID-0410, DEPT-NAME-0410, and DEPT-HEAD-ID-0410 are grouped together in the DEPARTMENT record because they all store information about a department. The DEPARTMENT record is called a **database record** because it describes information in the database. Database records are defined by using the schema compiler. (Alternatively, in an SQL environment, tables are defined by using SQL.) DBAs usually are responsible for defining and maintaining database records.

You also can define **work records** for an application. Work records describe temporary storage for a dialog. Elements in a work record describe data that is not stored in the application database. You define work records by using the **IDD menu facility**.

Note: Work records are sometimes referred to as **IDD records** to differentiate them from schema-defined database records.

Database and work records must be associated with each dialog that uses them.

Associating database records with a dialog: You associate database records with a dialog when you add a subschema containing the records to the dialog.

For example, subschema EMPSS01 contains the DEPARTMENT record. When you added subschema EMPSS01 to dialog *XXX*DADD in 12.4.3, “Step 3: Add a subschema” on page 12-24, you made the DEPARTMENT record available to that dialog.

Associating work records with a dialog: You associate work records with a dialog either by naming the record when you define the dialog's map or when you define the dialog itself, depending on how the work record is to be used:

- **If the work record is to be displayed on the dialog's map** (and optionally used in the dialog's process logic), you add the record with the map.

  Work records named on a map are automatically available to dialogs that use the map.

- **If the work record is to be used only by process module commands** (and not displayed on the dialog's map), you add the work record directly to the dialog.

Note that both database and work records that define data to be displayed on a map must be directly associated with the map. Because of this, records on a map are often referred to collectively as **map records**.

Note: **Work records** associated with a map are automatically available to dialogs that use the map. For **database records** associated with a map, you still must give the dialog a subschema to make the records available to the dialog. This is because information about the database itself is defined in the subschema, and is required by dialogs that access database records.

In this chapter, you will define a work record for dialog *XXX*DUPD. Data for the record will be used internally by process modules but will not be displayed to the end user. Therefore, you will associate this record directly with dialog *XXX*DUPD (rather than with map *XXX*MAP) when you add process logic to the dialog in Chapter 15, “Completing the Department Application” on page 15-1.

14.3 Instructions

Work records typically are small and contain only elements needed by a particular dialog or process module. This reduces the amount of storage reserved for a given dialog during execution. For example, in this chapter you will define a one-element work record for dialog *XXX*DUPD.

You will use the **IDD menu facility** to define:

- **Element** *XXX***-WK-FIRST-TIME** — An element for use in response process *XXX*DUPD-ENTER.

  Element *XXX*-WK-FIRST-TIME stores either Y (yes) or N (no) to indicate whether response process *XXX*DUPD-ENTER is processing a department record for the first time, or not. Different commands in process module *XXX*DUPD-ENTER are performed, based on the value in element *XXX*-WK-FIRST-TIME.
- **Record** *XXX***-WK-RECORD** — A work record for dialog *XXX*DUPD. Record *XXX*-WK-RECORD contains element *XXX*-WK-FIRST-TIME.

Additional elements can be added to a work record at any time. For example, if a response process for dialog *XXX*DUPD is modified so that a counter is required, you can add an element (for example, *XXX*-WK-COUNTER) to record *XXX*-WK-RECORD to store the counter value.

To define a work record in this chapter, you will:

1. Define element *XXX*-WK-FIRST-TIME.
2. Define work record *XXX*-WK-RECORD.

These steps are presented below. If you need additional information at any time about the use of IDD, see B.6, “Using the IDD Menu Facility” on page B-31.

14.3.1 Step 1: Define an element

Each **element** that you add to the data dictionary describes a unit of data that can be used in an application.

In order to define an element, you must invoke and sign on to the IDD menu facility. For example, when using IDMS-DC, you invoke the IDD menu facility as shown:

```
ENTER NEXT TASK CODE:
iddmt
                                                                  [Enter]
```

For more information on invoking and signing on to the IDD menu facility, see 12.3.1, “Step 1: Invoke the IDD menu facility” on page 12-10.

You use the **Element Entity screen** to define elements in the data dictionary. You display the Element Entity screen as shown:

```
-> elem
                                                                        [Enter]
```

Element Entity screen

```
     IDD REL 15.0                *** ELEMENT ENTITY ***                  ELEM
->
                                 DICT=DEMO

X DISPLAY      ELEMENT NAME....:
_ MODIFY
_ ADD          VERSION NUMBER..: 1       _ HIGHEST     _ NEXT HIGHEST
_ DELETE                                 _ LOWEST      _ NEXT LOWEST

DESCRIPTION:

PICTURE....:                                     NO SYNC: X   SYNC: _

USAGE......: X DISPLAY                   _ CONDITION NAME (LEVEL 88)
             _ COMP/COMP-4 (BINARY)      _ COMP-3 (PACKED DECIMAL)
             _ COMP-1 (SHORT FLOATING)   _ COMP-2 (LONG FLOATING)
             _ BIT                       _ POINTER

_ ELMX  =  ELEMENT EXTENSION <PF9>          _ SUBE  =  SUBORD ELEMENTS   <PF11>
_ REGN  =  USER REGISTRATION <PF2>          _ PUBL  =  PUBLIC ACCESS     <PF3>
_ CLAT  =  CLASS/ATTRIBUTES  <PF4>          _ RKEY  =  RELATIONAL KEYS   <PF5>
_ COMM  =  COMMENTS          <PF6>          _ COML  =  COMMENT KEY LIST  <PF7>
_ HIST  =  HISTORY           <PF8>          _ COPY  =  SAME AS/COPY FROM
_ XREF  =  CROSS REFERENCE   <PF10>         _ HELP  =  HELP              <PF1>
```

Screen prompts: When you use the Element Entity screen to define an element, you typically enter specifications for the following Element Entity screen prompts:

- **ELEMENT NAME** — You must supply an element name. The name you specify must be unique.
- **DISPLAY** — You deselect the DISPLAY action when you intend to add a new element.

  To do this, type a blank over the X displayed to the left of the action.
- **ADD** — You must select the ADD action to specify that you are defining a new element.
- **DESCRIPTION** — You optionally specify a brief description of the element.
- **PICTURE** — You must specify the storage layout (that is, **picture**) for the element after the PICTURE prompt. A picture specifies:
  - The **type of data** that can be stored for the element:
    - For **alphanumeric** data, enter an X.
    - For **alphabetic** data, enter an A.
    - For **numeric** data, enter a 9.
  - The **number of characters** that can be stored for the element.

    By default, an element can store **single-character values**.

You optionally enable an element to store **larger values** by specifying, in parentheses, the maximum number of characters to be stored by the element. For example, a picture of X(6) enables an element to store values that contain up to six alphanumeric characters.

For example, the element that you define in this chapter will be used to store the single-character value Y or N.

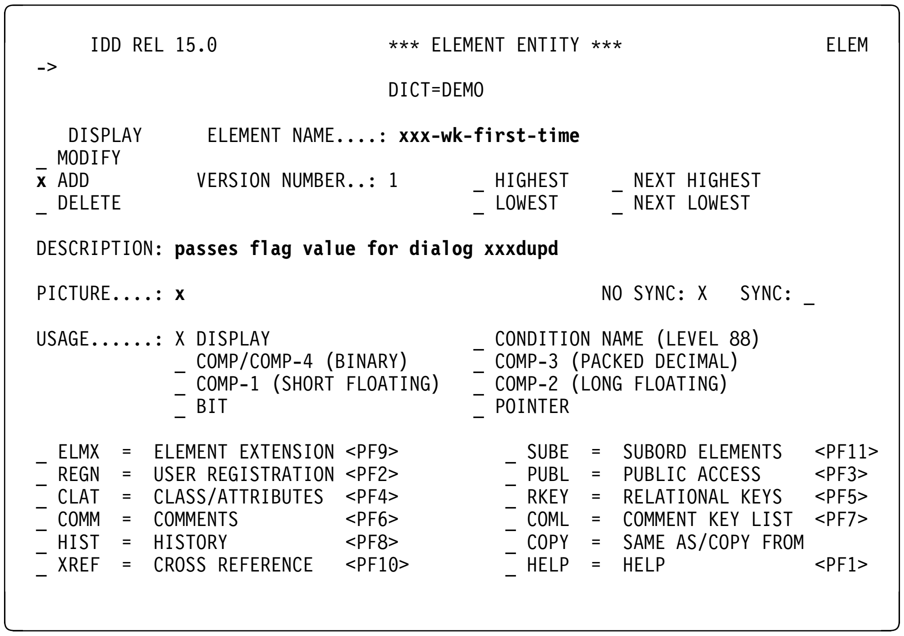
- **USAGE** — You optionally specify the storage format for data after the USAGE prompt.

  For element *XXX*-WK-FIRST-TIME, you will retain the default usage, **DISPLAY**, since DISPLAY usage is appropriate for the flag values (Y and N) stored for the element.

  For more information on DISPLAY and other USAGE specifications, see the *IDD DDDL Reference*.

Defining an element: To define *XXX*-WK-FIRST-TIME as an element that stores single-character alphanumeric values, you use the Element Entity screen to:

- Type the name of the element. You can use your initials instead of *XXX*
- Deselect the DISPLAY action and select the ADD action.
- Type an optional description for the element.
- Specify a picture of X. This enables the element to store single-character alphanumeric flag values.



```
     IDD REL 15.0               *** ELEMENT ENTITY ***                  ELEM
->
                                DICT=DEMO

   DISPLAY      ELEMENT NAME....: xxx-wk-first-time
_ MODIFY
x ADD          VERSION NUMBER..: 1        _ HIGHEST     _ NEXT HIGHEST
_ DELETE                                  _ LOWEST      _ NEXT LOWEST

DESCRIPTION: passes flag value for dialog xxxdupd

PICTURE....: x                                     NO SYNC: X   SYNC: _

USAGE......: X DISPLAY                    _ CONDITION NAME (LEVEL 88)
             _ COMP/COMP-4 (BINARY)       _ COMP-3 (PACKED DECIMAL)
             _ COMP-1 (SHORT FLOATING)    _ COMP-2 (LONG FLOATING)
             _ BIT                        _ POINTER

_ ELMX  =  ELEMENT EXTENSION <PF9>           _ SUBE  =  SUBORD ELEMENTS   <PF11>
_ REGN  =  USER REGISTRATION <PF2>           _ PUBL  =  PUBLIC ACCESS     <PF3>
_ CLAT  =  CLASS/ATTRIBUTES  <PF4>           _ RKEY  =  RELATIONAL KEYS   <PF5>
_ COMM  =  COMMENTS          <PF6>           _ COML  =  COMMENT KEY LIST  <PF7>
_ HIST  =  HISTORY           <PF8>           _ COPY  =  SAME AS/COPY FROM
_ XREF  =  CROSS REFERENCE   <PF10>          _ HELP  =  HELP              <PF1>
```

After you press [Enter], the Element Entity screen displays a message to indicate whether the element definition has been added to the data dictionary:

- **If you successfully defined a new element**, IDD displays a message like:

  ```
  ELEMENT 'XXX-WK-FIRST-TIME' VERSION 1 ADDED
  ```

- **If the element cannot be added to the data dictionary**, IDD displays a different message.

  Read the message to determine the problem. You can type over any errors and press [Enter] again.

After you successfully add element *XXX*-WK-FIRST-TIME to the data dictionary, you can define work record *XXX*-WK-RECORD.

14.3.2 Step 2: Define a work record

You define a **record** to describe a collection of one or more existing elements. To define a work record to the data dictionary, you perform the following steps using the IDD menu facility:

1. **You specify basic information for the record** by using the Record Entity screen.
2. **You add one or more existing elements to the record** by using the Record Element screen. In this chapter, you will use this screen to add element *XXX*-WK-FIRST-TIME to record *XXX*-WK-RECORD.

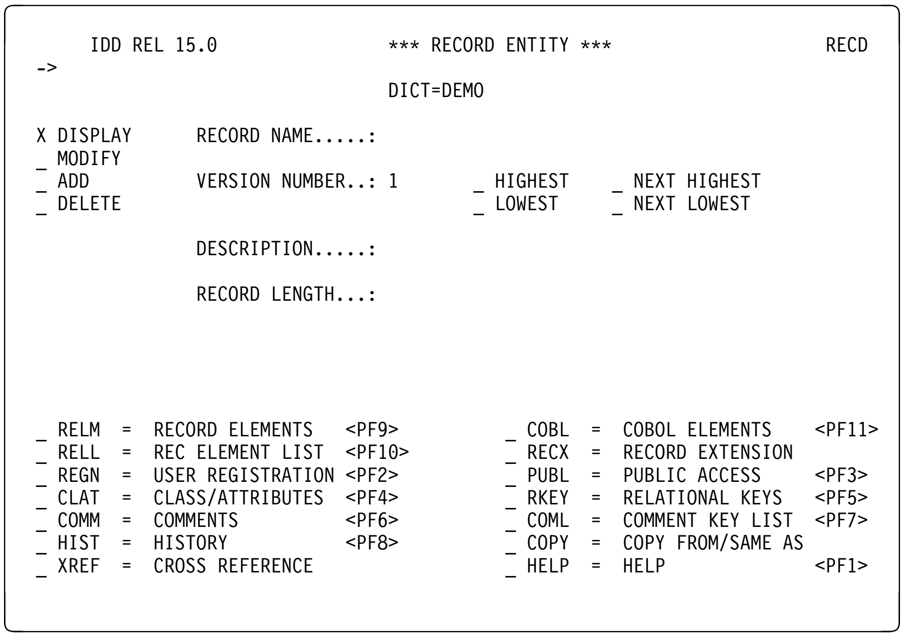
The way you specify basic information for a work record and add elements to the record are shown below.

14.3.3 Step 3: Specifying basic information

You use the **Record Entity** screen to define a work record. You display the Record Entity screen as shown:

```
-> recd
                                                                    [Enter]
```

Record Entity screen: The **Record Entity** screen is displayed.



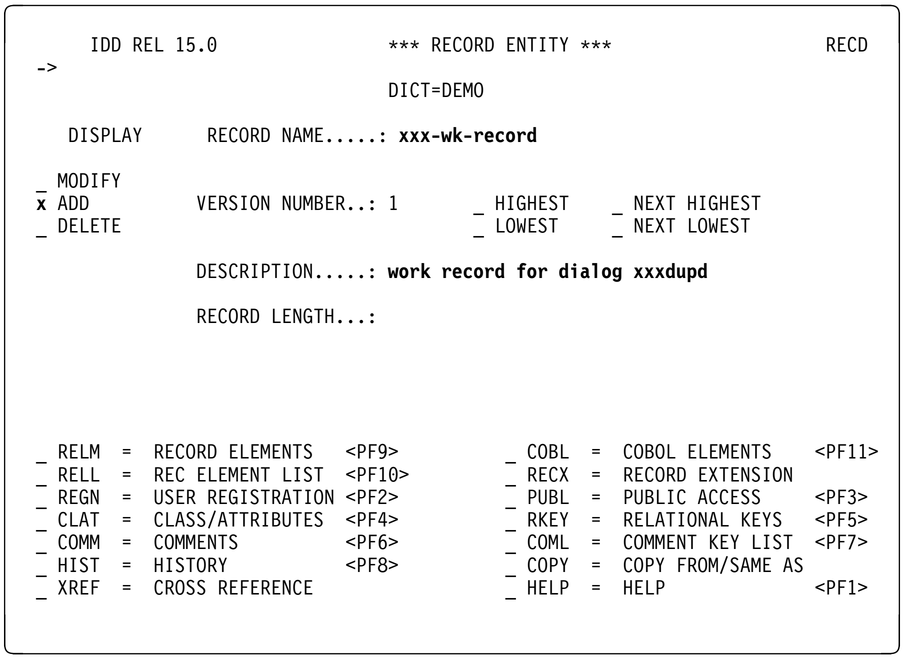
```
     IDD REL 15.0                *** RECORD ENTITY ***                   RECD
->
                                 DICT=DEMO

X DISPLAY      RECORD NAME.....:
_ MODIFY
_ ADD          VERSION NUMBER..: 1       _ HIGHEST    _ NEXT HIGHEST
_ DELETE                                 _ LOWEST     _ NEXT LOWEST

               DESCRIPTION.....:

               RECORD LENGTH...:




_ RELM  =  RECORD ELEMENTS   <PF9>         _ COBL  =  COBOL ELEMENTS    <PF11>
_ RELL  =  REC ELEMENT LIST  <PF10>        _ RECX  =  RECORD EXTENSION
_ REGN  =  USER REGISTRATION <PF2>         _ PUBL  =  PUBLIC ACCESS     <PF3>
_ CLAT  =  CLASS/ATTRIBUTES  <PF4>         _ RKEY  =  RELATIONAL KEYS   <PF5>
_ COMM  =  COMMENTS          <PF6>         _ COML  =  COMMENT KEY LIST  <PF7>
_ HIST  =  HISTORY           <PF8>         _ COPY  =  COPY FROM/SAME AS
_ XREF  =  CROSS REFERENCE                 _ HELP  =  HELP              <PF1>
```

Screen prompts: When adding a new work record, you usually specify information for the following Record Entity screen prompts:

- **RECORD NAME** — You must supply a record name. The name that you specify must be unique.
- **DISPLAY** — You deselect the DISPLAY action when you intend to add a new record.
- **ADD** — You select the ADD action to specify that you are defining a new record.
- **DESCRIPTION** — You optionally provide a brief description of the record.

To specify basic information for record *XXX*-WK-RECORD. You use the Record Entity screen and enter the indicated specifications:

- Type the name of the record You can use your initials instead of *XXX*
- Deselect the DISPLAY action and select the ADD action.
- Type an optional description for the record



```
    IDD REL 15.0               *** RECORD ENTITY ***                   RECD
->
                               DICT=DEMO

   DISPLAY      RECORD NAME.....: xxx-wk-record

_ MODIFY
x ADD          VERSION NUMBER..: 1       _ HIGHEST    _ NEXT HIGHEST
_ DELETE                                 _ LOWEST     _ NEXT LOWEST

               DESCRIPTION.....: work record for dialog xxxdupd

               RECORD LENGTH...:




_ RELM  =  RECORD ELEMENTS   <PF9>          _ COBL  =  COBOL ELEMENTS    <PF11>
_ RELL  =  REC ELEMENT LIST  <PF10>         _ RECX  =  RECORD EXTENSION
_ REGN  =  USER REGISTRATION <PF2>          _ PUBL  =  PUBLIC ACCESS     <PF3>
_ CLAT  =  CLASS/ATTRIBUTES  <PF4>          _ RKEY  =  RELATIONAL KEYS   <PF5>
_ COMM  =  COMMENTS          <PF6>          _ COML  =  COMMENT KEY LIST  <PF7>
_ HIST  =  HISTORY           <PF8>          _ COPY  =  COPY FROM/SAME AS
_ XREF  =  CROSS REFERENCE                  _ HELP  =  HELP              <PF1>
```

After you press [Enter], the Record Entity screen displays a message to inform you whether the record definition has been stored in the data dictionary:

- **If you successfully defined a new record**, IDD displays a message like:

  ```
  RECORD 'XXX-WK-RECORD' VERSION 1 ADDED
  ```

- **If the record cannot be added to the data dictionary**, IDD displays a different message.

  In this case, read the message to determine the problem. You can change specifications on the Record Entity screen and press [Enter] again.

After you successfully specify basic information for a record, you can add elements to the record by using the Record Element screen.

14.3.4 Adding elements

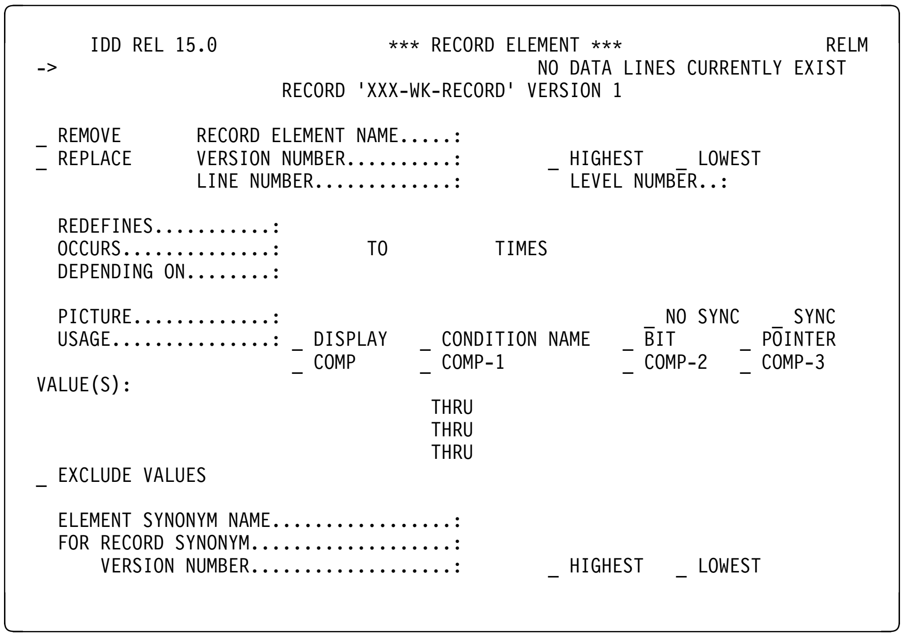
You defined element *XXX*-WK-FIRST-TIME in the data dictionary in 14.3.1, “Step 1: Define an element” on page 14-6 earlier in this chapter. In this step, you will add element *XXX*-WK-FIRST-TIME to record *XXX*-WK-RECORD. Elements used in records are referred to as **record elements**.

You use the IDD **Record Element screen** to add elements to work records:

```
-> relm
                                                                   [Enter]
```

RELM identifies the Record Element screen.

The Record Element screen is displayed.



```
     IDD REL 15.0                *** RECORD ELEMENT ***                  RELM
->                                             NO DATA LINES CURRENTLY EXIST
                       RECORD 'XXX-WK-RECORD' VERSION 1

_ REMOVE      RECORD ELEMENT NAME.....:
_ REPLACE     VERSION NUMBER..........:        _ HIGHEST    _ LOWEST
              LINE NUMBER.............:          LEVEL NUMBER..:

  REDEFINES...........:
  OCCURS..............:       TO          TIMES
  DEPENDING ON........:

  PICTURE.............:                                  _ NO SYNC    _ SYNC
  USAGE...............: _ DISPLAY   _ CONDITION NAME   _ BIT       _ POINTER
                        _ COMP      _ COMP-1           _ COMP-2    _ COMP-3
VALUE(S):
                                     THRU
                                     THRU
                                     THRU
_ EXCLUDE VALUES

  ELEMENT SYNONYM NAME.................:
  FOR RECORD SYNONYM...................:
      VERSION NUMBER...................:        _ HIGHEST    _ LOWEST
```

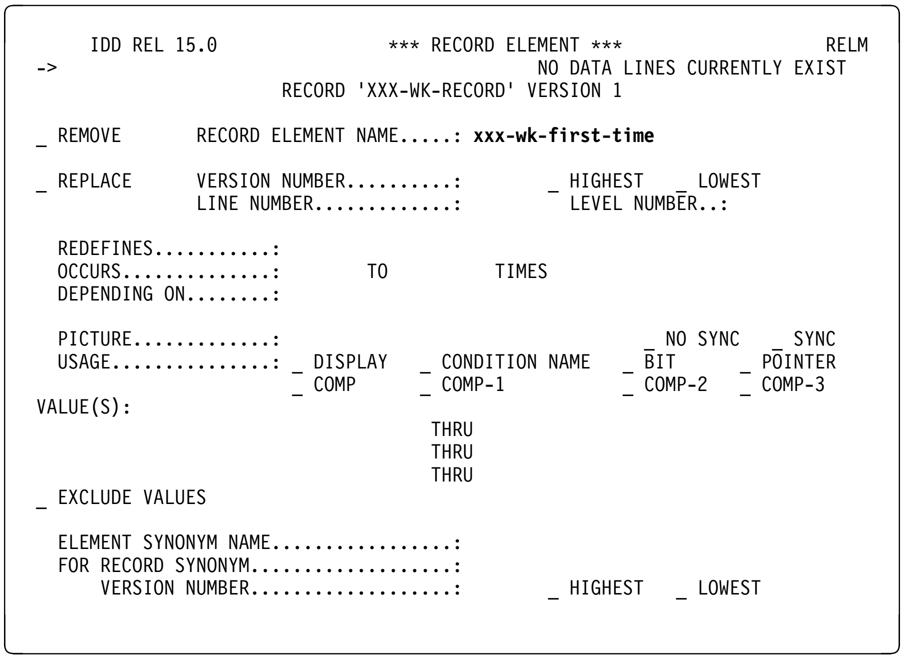
Screen prompts: When you add an element to a record, you typically specify the following information:

- **Element name** — You must name an existing element after the RECORD ELEMENT NAME prompt.
- **Additional specifications** — You optionally can redefine how the element is used in the current record by using prompts on the Record Element screen. For example, you could override the **PICTURE** or **USAGE** specifications you gave to element *XXX*-WK-FIRST-TIME in 14.3.1, "Step 1: Define an element" on page 14-6 earlier in this chapter.

  You most often override specifications for an element when the element is used in different ways by several work records. Specifications made on the Record Element screen do not alter the actual element definition in the data dictionary. For more information on Record Element screen specifications, see the *IDD DDDL Reference*.

  You will not change any values when you add element *XXX*-WK-FIRST-TIME to record *XXX*-WK-RECORD in this chapter.

You use the Record Element screen to type in the name of the element that you defined earlier in this chapter.



```
     IDD REL 15.0                *** RECORD ELEMENT ***                  RELM
->                                           NO DATA LINES CURRENTLY EXIST
                      RECORD 'XXX-WK-RECORD' VERSION 1

_ REMOVE       RECORD ELEMENT NAME.....: xxx-wk-first-time

_ REPLACE      VERSION NUMBER..........:        _ HIGHEST    _ LOWEST
               LINE NUMBER.............:         LEVEL NUMBER..:

  REDEFINES...........:
  OCCURS..............:        TO          TIMES
  DEPENDING ON........:

  PICTURE.............:                                 _ NO SYNC    _ SYNC
  USAGE...............: _ DISPLAY    _ CONDITION NAME   _ BIT       _ POINTER
                        _ COMP       _ COMP-1           _ COMP-2    _ COMP-3
VALUE(S):
                                      THRU
                                      THRU
                                      THRU
_ EXCLUDE VALUES

  ELEMENT SYNONYM NAME.................:
  FOR RECORD SYNONYM...................:
      VERSION NUMBER...................:        _ HIGHEST    _ LOWEST
```

After you press [Enter], the Record Element screen displays a message to indicate whether the element has been added to the work record in the data dictionary:

- **If the element is successfully added to the record**, IDD displays a message like:

  ```
  RECORD 'XXX-WK-RECORD' VERSION 1 MODIFIED
  ```

- **If the element cannot be added to the record**, IDD displays a different message than the one shown above.

  In this case, read the message to determine the problem. You can type over any errors and press [Enter] again.

After you finish defining work elements and records, you can exit from IDD. In this sample session, exiting from IDD is optional because you will use the IDD menu facility again in the next chapter.

If you want to exit from IDD, enter the SWITCH SUSPEND command in the command area of any IDD menu facility screen:

```
-> switch suspend
                                                [Enter]
```

14.4 Summary

You can define a work record to establish temporary storage for a dialog. Elements in the work record describe the data to be stored.

In this chapter, you used the IDD menu facility to create a work record for use in dialog XXXDUPD by performing the following steps:

1. **You defined element** *XXX*-**WK-FIRST-TIME** to store a single-character, alphanumeric status value. When used in dialog *XXX*DUPD, this element contains either Y (yes) or N (no). This value establishes whether response process *XXX*DUPD-ENTER is processing a department for the first time or not.

2. **You defined record** *XXX*-**WK-RECORD** to contain element *XXX*-WK-FIRST-TIME.

To see how you use work records and elements in dialogs, proceed to the next chapter, chapter 10, where you enhance dialog XXXDUPD by adding process modules and data definitions to the dialog.

Chapter 15. Completing the Department Application

15.1 Introduction

When you created the prototype of the Department application, you defined skeleton dialogs *XXX*DADD and *XXX*DUPD to be displayed by dialog functions in the application. In previous chapters, you completed dialog *XXX*DADD by adding modules of process commands for the dialog.

In this chapter, you will complete the sample Department application by adding process logic to dialog *XXX*DUPD. The process modules that you define for the dialog will allow the end user to modify and delete existing department information.

This chapter includes:

- An overview of process modules and dialog execution
- Steps for defining process modules
- Steps for associating process modules with dialogs
- Steps for executing the application
- A summary of what you've accomplished in this manual

15.2 Overview

The final structure of the sample Department application is shown below. Dialog *XXX*DUPD is the only component in the application that requires further development. In this chapter, you will enhance dialog *XXX*DUPD by writing three process modules for the dialog. When you complete dialog *XXX*DUPD, you will have finished defining the sample Department application. Dialog XXXDUPD, which is executed by functions MODDEP and DELDEP, is the only component in the sample application that requires further development.

As the diagram indicates, dialog *XXX*DUPD is executed by both functions MODDEP and DELDEP. Therefore, process modules that you write for dialog *XXX*DUPD must be able to both modify and delete department information in the database.

To handle all processing requirements for dialog *XXX*DUPD, you will define the following premap and response processes for the dialog:

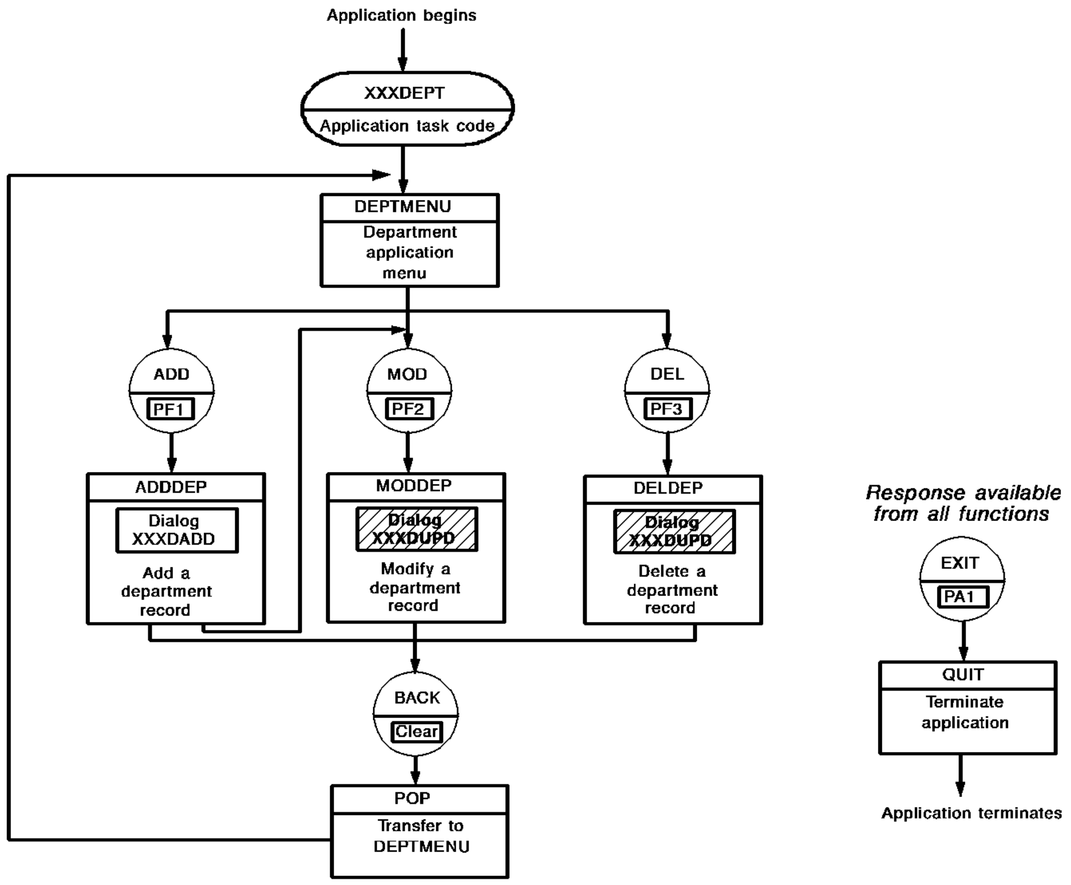
Process module	**Type**	**Function performed**
*XXX*DUPD-PREMAP	Premap	Displays the dialog's map with a message prompting the end user for the department to be modified or deleted.
*XXX*DUPD-ENTER	Response (associated with the [Enter] key)	Handles most modification and deletion operations.
*XXX*DUPD-PA2	Response (associated with the [PA2] key)	Allows the end user to cancel the current modification or deletion operation before the database is updated.

When you add these process modules to dialog *XXX*DUPD, you also must enable the process modules to access temporary storage and database information. To do this, you will associate with the dialog any records that define temporary storage and database records. You will add:

1. **Work record *XXX*-WK-RECORD** — Includes element *XXX*-WK-FIRST-TIME, which stores a status value (Y or N) used by response processes in dialog *XXX*DUPD.

   You defined work record *XXX*-WK-RECORD in Chapter 14, “Defining Work Records Using IDD” on page 14-1.

2. **Subschema EMPSS01** — Includes the DEPARTMENT record, which is the database record for department information.



At run time, the dialog can access any database record in the dialog's subschema. Database administrators (DBAs) typically define subschemas for use by dialogs.

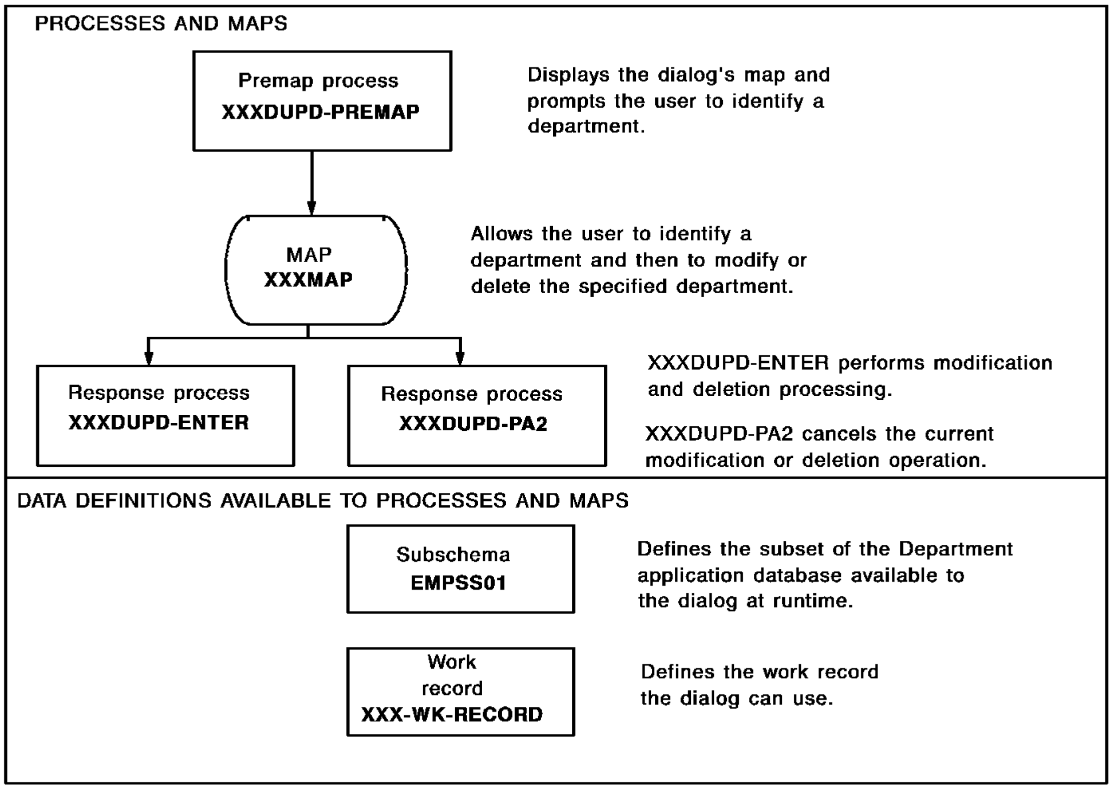
The following diagram shows dialog *XXX*DUPD with all of its components. To enable dialog XXXDUPD to perform all necessary processing, you will add a premap process, two response processes, a work record, and a subschema to the dialog.

Cancelling a modification: To protect data in the database, your process modules should make it easy for end users to cancel a modification or deletion operation. As an application developer, you will do this by defining a **two-stage procedure** for modifying or deleting department records:

1. **First, the end user specifies a department to modify or delete**. When the end user presses [Enter] to specify a department, response process *XXX*DUPD-ENTE is executed. This response process accesses and displays the complete department record.

2. **Then, the end user either cancels or continues the current operation**, based on the record displayed on the screen:

   - **To cancel**, the end user presses [PA2].


PROCESSES AND MAPS
Premap process XXXDUPD-PREMAP
Displays the dialog's map and prompts the user to identify a department.
MAP XXXMAP
Allows the user to identify a department and then to modify or delete the specified department.
Response process XXXDUPD-ENTER
Response process XXXDUPD-PA2
XXXDUPD-ENTER performs modification and deletion processing.
XXXDUPD-PA2 cancels the current modification or deletion operation.
DATA DEFINITIONS AVAILABLE TO PROCESSES AND MAPS
Subschema EMPSS01
Defines the subset of the Department application database available to the dialog at runtime.
Work record XXX-WK-RECORD
Defines the work record the dialog can use.

- **To continue**, the end user types any necessary modifications to the record and presses [Enter]. This time, response process *XXX*DUPD-ENTER modifies or deletes the department record in the database.

Response process *XXX*DUPD-ENTER performs both stages of a completed modification or deletion operation. This is accomplished at run time by having the status value (Y or N) in element *XXX*-WK-FIRST-TIME determine which stage is performed. Commands in your process modules change the value in *XXX*-WK-FIRST-TIME, as appropriate.

Instructions for defining process modules, adding the modules to a dialog, and executing the final application are given on the following pages.

15.3 Defining process modules using IDD

According to your application definition, dialog *XXX*DUPD is executed by functions MODDEP and DELDEP:

- Function **MODDEP** executes dialog *XXX*DUPD to **modify** department records.
- Function **DELDEP** executes dialog *XXX*DUPD to **delete** department records.

Process modules that you write must be able to handle both modification and deletion operations. To accomplish this, you will include commands that:

1. Test which function (MODDEP or DELDEP) is currently in use
2. Invoke subroutines appropriate to the current function

You test which function is currently in use by querying **AGR-CURRENT-FUNCTION**, which is an element in the system-supplied ADSO-APPLICATION-GLOBAL-RECORD. At run time, AGR-CURRENT-FUNCTION stores the name of the current function. Your process module can access AGR-CURRENT-FUNCTION because ADSO-APPLICATION-GLOBAL-RECORD belongs to the dialog's map (and thus to the dialog).

For example, the following conditional command tests the value in AGR-CURRENT-FUNCTION:

```
IF AGR-CURRENT-FUNCTION EQ 'MODDEP'    ◄-- If function MODDEP is executing
THEN                                      the dialog,

    CALL MODRTN.                       ◄-- call subroutine MODRTN to
                                          modify the department record.
```

In this chapter, you will define process modules for dialog *XXX*DUPD. You will:

1. Define process module *XXX*DUPD-PREMAP.
2. Define process module *XXX*DUPD-ENTER.
3. Define process module *XXX*DUPD-PA2.

15.3.1 Step 1: Define process module XXXDUPD-PREMAP

In order to define a process module, you must be using IDD. If you *did not* remain signed on to IDD at the end of the previous chapter, you should now sign on to the IDD menu facility, as described in 12.3.1, “Step 1: Invoke the IDD menu facility” on page 12-10.

The first process module you will define in this chapter is process module *XXX*DUPD-PREMAP. Statements that you will input for the process module are shown below. This process module will be the premap process for dialog XXXDUPD.

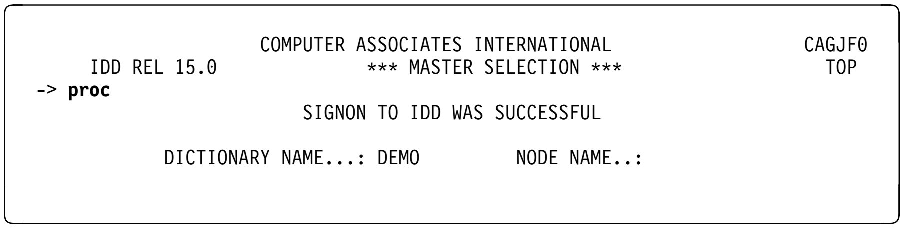
```
MOVE 'Y' TO XXX-WK-FIRST-TIME.                                                  1

IF AGR-CURRENT-FUNCTION EQ 'MODDEP'                                          -┐
THEN                                                                          │  2
    DISPLAY MSG TEXT                                                          │
        'MODIFY -- ENTER THE DEPARTMENT ID, OR SELECT: BACK OR EXIT'.-┘

ELSE                                                                         -┐
    DISPLAY MSG TEXT                                                          │  3
        'DELETE -- ENTER THE DEPARTMENT ID, OR SELECT: BACK OR EXIT'.-┘
```

1 This statement sets the flag value in *XXX*-WK-FIRST-TIME to Y (YES).

2 If the MODDEP function is in use, the dialog's map is displayed with the MODIFY message (defined here between the single quotation marks).

3 If the DELDEP function is in use, the dialog's map is displayed with the DELETE message.

Specifying basic information about the process module: You use the **Process Entity** screen to specify basic information for a process module. You can display and use the Process Entity screen as shown:



```
                   COMPUTER ASSOCIATES INTERNATIONAL                CAGJF0
    IDD REL 15.0               *** MASTER SELECTION ***               TOP
 -> proc
                       SIGNON TO IDD WAS SUCCESSFUL

           DICTIONARY NAME...: DEMO         NODE NAME..:
```

[Enter]

Enter the indicated specifications:

- Type the name of the process module. You can use your initials instead of *XXX*.
- Deselect the DISPLAY action and select the ADD action.
- Optionally type a description of the process.

```
    IDD REL 15.0               *** PROCESS ENTITY ***                  PROC
->
                               DICT=DEMO

  DISPLAY      PROCESS NAME....: xxxdupd-premap
_ MODIFY
x ADD         VERSION NUMBER..: 1       _ HIGHEST     _ NEXT HIGHEST
_ DELETE                                _ LOWEST      _ NEXT LOWEST

              DESCRIPTION.....: display map to mod/del departments
```

When you press [Enter], IDD redisplays the Process Entity screen with a message. If the process module is defined successfully, the Process Entity screen displays a message like:

```
PROCESS 'XXXDUPD-PREMAP' VERSION 1 ADDED
```

If a different message is displayed, read the message to determine the problem. You can type over any errors, and then press [Enter] again.

After you specify basic information about a process module, you can use the **Process Source screen** to enter process commands for the process module. For example, enter process commands for process module *XXX*DUPD-PREMAP as shown:

```
-> srcs
                                                    [Enter]
```

SRCE identifies the Process Source screen.

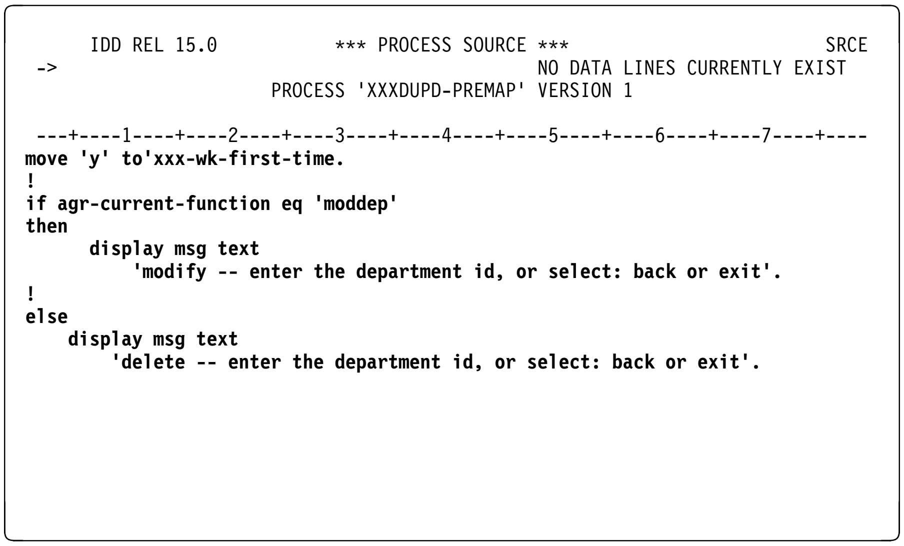
The Process Source screen is displayed.

```
    IDD REL 15.0           *** PROCESS SOURCE ***                      SRCE
->                                       NO DATA LINES CURRENTLY EXIST
                    PROCESS 'XXXDUPD-PREMAP' VERSION 1

---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----
```

Entering process statements: Enter source statements for the process module:

- Enter keywords, periods, and single quotes as shown.
- The exclamation point in any column signals the start of a comment.
- You can types spaces to indent statements, making the process source easier to read and debug.

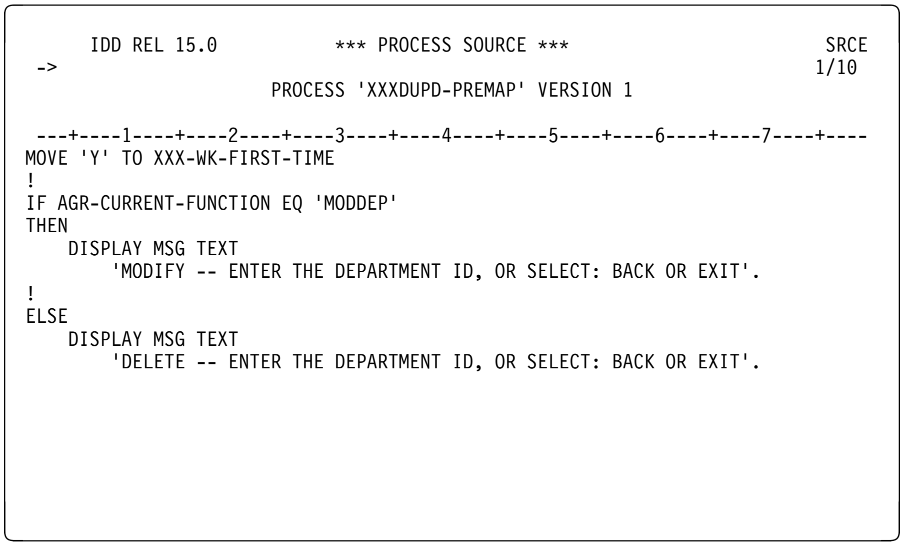
Caution: Don't type any characters beyond column 72.



```
      IDD REL 15.0           *** PROCESS SOURCE ***                        SRCE
 ->                                            NO DATA LINES CURRENTLY EXIST
                        PROCESS 'XXXDUPD-PREMAP' VERSION 1

 ---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----
move 'y' to'xxx-wk-first-time.
!
if agr-current-function eq 'moddep'
then
      display msg text
          'modify -- enter the department id, or select: back or exit'.
!
else
    display msg text
        'delete -- enter the department id, or select: back or exit'.
```

[Enter]

The Process Source screen is redisplayed.



```
      IDD REL 15.0           *** PROCESS SOURCE ***                        SRCE
 ->                                                                        1/10
                        PROCESS 'XXXDUPD-PREMAP' VERSION 1

 ---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----
MOVE 'Y' TO XXX-WK-FIRST-TIME
!
IF AGR-CURRENT-FUNCTION EQ 'MODDEP'
THEN
    DISPLAY MSG TEXT
        'MODIFY -- ENTER THE DEPARTMENT ID, OR SELECT: BACK OR EXIT'.
!
ELSE
    DISPLAY MSG TEXT
        'DELETE -- ENTER THE DEPARTMENT ID, OR SELECT: BACK OR EXIT'.
```

After you press [Enter], IDD adds the process module to the data dictionary and redisplays the Process Source screen with a message like:

```
PROCESS 'XXXDUPD-PREMAP' VERSION 1 MODIFIED
```

After you finish using the Process Source screen for process module *XXX*DUPD-PREMAP, you can proceed to define process module *XXX*DUPD-ENTER.

15.3.2 Step 2: Define process module XXXDUPD-ENTER

Now you will define process module *XXX*DUPD-ENTER for dialog *XXX*DUPD. This process module will be executed when the end user presses [Enter].

Process *XXX*DUPD-ENTER allows an end user to specify a department, verify that the correct department is specified, and then enter modification or deletion instructions by performing the following processing:

1. **When the end user first specifies the department to be processed and presses [Enter]**, response process *XXX*DUPD-ENTER:
   - Retrieves information stored for the specified department
   - Displays the department information for verification and prompts the end user to modify or delete the displayed record
2. **When the end user presses [Enter] again**, response process *XXX*DUPD-ENTER:
   - Modifies or deletes the department in the database
   - Displays the dialog's map to the end user with a message confirming the modification or deletion operation

To enable process module *XXX*DUPD-ENTER to perform the above processing, commands in the process module store and evaluate a status value in element *XXX*-WK-FIRST-TIME:

- **A value of Y (yes)** indicates that the process module is executing for the first time for a given department, and so must retrieve the department from the database.

  Before redisplaying the screen to the end user, the response process changes the value in element *XXX*-WK-FIRST-TIME from Y to N.

- **A value of N (no)** indicates that the process module is executing for the second time for the department, and so must modify or delete information stored for the department.

  After processing the department, the response process resets the value in element *XXX*-WK-FIRST-TIME from N to Y.

Process module *XXX*DUPD-ENTER is shown below. Statements that retrieve the specified department from the database, modify the department, and delete the department are organized into subroutines.

```
READY USAGE-MODE UPDATE.
                                                                  ─┐
IF AGR-CURRENT-RESPONSE NE SPACES                                  │
AND NO FIELDS CHANGED                                              │
AND XXX-WK-FIRST-TIME EQ 'Y'                                       │ 1
THEN                                                               │
   EXECUTE NEXT FUNCTION.                                         ─┘

IF XXX-WK-FIRST-TIME EQ 'Y'                                       ─┐
THEN                                                               │ 2
```

```
    CALL FRSTRTN.                                                        -┘

IF AGR-CURRENT-FUNCTION EQ 'MODDEP'                                      -┐
THEN
    CALL MODRTN.                                                            3
ELSE
    CALL DELRTN.                                                         -┘

DEFINE FRSTRTN.                                                           --  4

OBTAIN CALC DEPARTMENT.                                                   --  5

IF DB-REC-NOT-FOUND                                                       -┐
THEN
    DISPLAY MSG TEXT                                                          6
        'DEPARTMENT DOES NOT EXIST--SPECIFY A DIFFERENT DEPARTMENT'.
                                                                          -┘
ELSE DO.                                                                  -┐
    MOVE 'N' TO XXX-WK-FIRST-TIME.                                            7
    PROTECT FIELD DEPT-ID-0410 TEMPORARY.                                 -┘


    IF AGR-CURRENT-FUNCTION EQ 'MODDEP'                                   -┐
    THEN
        DISPLAY MSG TEXT
            'MODIFY DEPARTMENT AND PRESS ENTER (PA2 TO CANCEL)'.
                                                                              8
    ELSE
        DISPLAY MSG TEXT
            'PRESS ENTER TO DELETE THIS DEPARTMENT (PA2 TO CANCEL)'.
                                                                          -┘
END.

DEFINE MODRTN.                                                             -- 9

MODIFY DEPARTMENT.                                                         -- 10

INITIALIZE (DEPARTMENT).                                                  -┐

MOVE 'Y' TO XXX-WK-FIRST-TIME                                                 11
DISPLAY MSG TEXT
    'DEPARTMENT MODIFIED--SPECIFY ANOTHER DEPARTMENT TO MODIFY'.          -┘


DEFINE DELRTN.                                                             -- 12

ERASE DEPARTMENT ALLOWING ('0230').                                        -- 13

MOVE 'Y' TO XXX-WK-FIRST-TIME.                                             -- 14

IF ERROR-STATUS EQ '0230'                                                 -┐
THEN
    DISPLAY MSG TEXT                                                          15
        'CANNOT DELETE THIS DEPARTMENT--SPECIFY ANOTHER DEPARTMENT'.
                                                                          -┘

ELSE DO.                                                                  -┐
    MOVE 'Y' TO XXX-WK-FIRST-TIME.
    INITIALIZE (DEPARTMENT).                                                   16
    DISPLAY MSG TEXT                                                  -┘
        'DEPARTMENT DELETED--SPECIFY ANOTHER DEPARTMENT TO DELETE'.

END.
```

1 Transfers control to another application function when the user enters a valid application response *and* doesn't try to input other information on the screen.

2 The first time this process is executed for a given department. subroutine FRSTRTN is called.

3 The next time the process is executed for the department, the appropriate subroutine is called (MODRTN for function MODDEP or DELRTN for function DELDEP).

FRSTRTN subroutine

4 Subroutine FRSTRTN begins.

5 Uses the department ID supplied by the user to retrieve the department.

6 If the specified department does not exist in the database, the dialog's map is redisplayed with the DEPARTMENT DOES NOT EXIST error message.

7 If the department does exist, the flag in *XXX*-WK-FIRST-TIME is set to N (NO) and the map field that displays department ID numbers is temporarily protected from user input.

8 The retrieved department record is displayed on the dialog's map with an appropriate message.

MODRTN subroutine

9 Subroutine MODRTN begins.

10 The department is modified in the database based on the user's input.

11 After the department is modified, dialog buffers for DEPARTMENT data are initialized, the flag in *XXX*-WK-FIRST-TIME is reset to Y (YES), and the map is redisplayed with a confirming message.

DELRTN subroutine

12 Subroutine DELRTN begins.

13 The department is deleted from the database. Your code tests for errors that cause error code 0230 (see below).

14 Resets the flag in *XXX*WK-FIRST-TIME to Y (YES).

15 If status code 0230 is returned, the department cannot be deleted because it owns other records (for example, employee records). In this case, the map is redisplayed with a message and the department is not deleted.

16 After the department is deleted, dialog buffers are initialized and the dialog's map is redisplayed with a confirming message.

Specifying basic information for the process module: To specify basic information for process module XXXDUPD-ENTER, you use the IDD Process Entity screen:

```
-> proc
                                                  [Enter]
```

Enter the screen identifier (PROC) in the command area.

Enter the indicated specifications on the Process Entity screen. Don't forget to deselect the DISPLAY action.

```
     IDD REL 15.0                *** PROCESS ENTITY ***                  PROC
->
                                 DICT=DEMO

   DISPLAY      PROCESS NAME....: xxxdupd-enter

_ MODIFY
x ADD          VERSION NUMBER..: 1       _ HIGHEST     _ NEXT HIGHEST
_ DELETE                                 _ LOWEST      _ NEXT LOWEST

               DESCRIPTION.....: retrieve dept and then mod/del dept
```

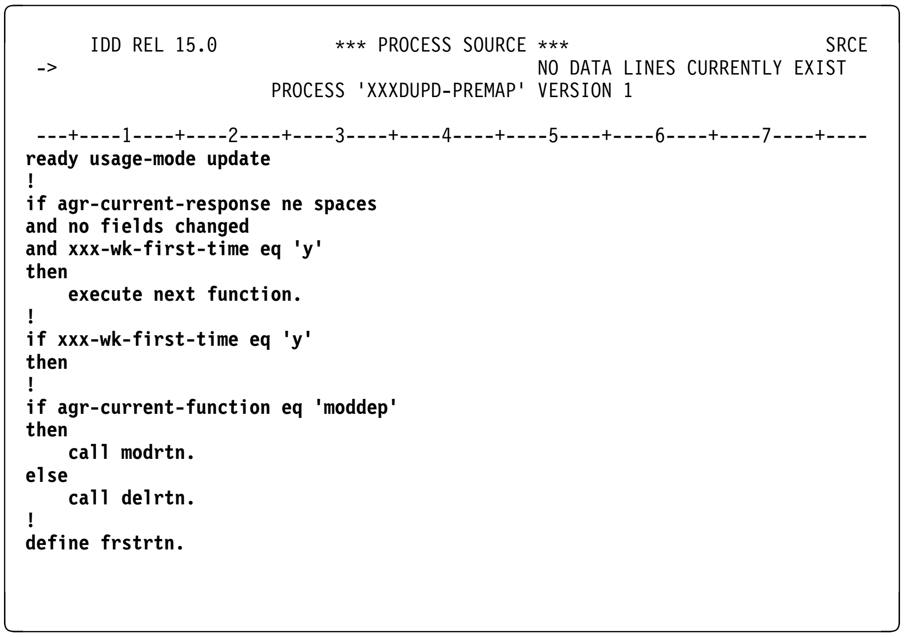
```
                                                  [Enter]
```

Entering source statements for the process: To enter source commands for a process module, you display and use the Process Source screen, as shown below for process *XXX*DUPD-ENTER:

```
-> srcs
                                                  [Enter]
```

SRCE identifies the Process Source screen.

The Process Source screen is displayed. Enter a full screen of source statements. Enter keywords, periods, and quotes as shown below. Don't type characters beyond column 72.



```
      IDD REL 15.0           *** PROCESS SOURCE ***                        SRCE
 ->                                        NO DATA LINES CURRENTLY EXIST
                         PROCESS 'XXXDUPD-PREMAP' VERSION 1

 ---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----
ready usage-mode update
!
if agr-current-response ne spaces
and no fields changed
and xxx-wk-first-time eq 'y'
then
    execute next function.
!
if xxx-wk-first-time eq 'y'
then
!
if agr-current-function eq 'moddep'
then
    call modrtn.
else
    call delrtn.
!
define frstrtn.
```

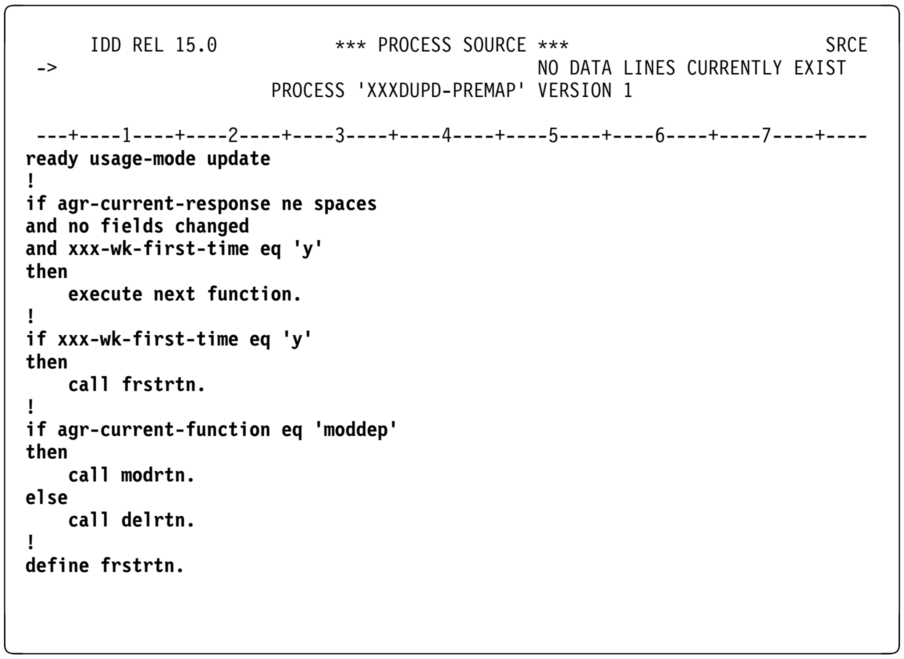
>> For more information on entering, moving, and deleting text on the Process Source screen, refer to *CA-IDMS Online Compiler Text Editor*.

After you type the first page of source commands for the process module, **open up new lines at the end of the Process Source screen** by performing the following steps:

1. **Move the cursor** to the line containing the final source command on the screen.
2. **Press [PF4]** (default) to insert new lines after the cursor.

You can press [PF5] (default) at any time to apply the new lines to the work file maintained by the IDD menu facility.

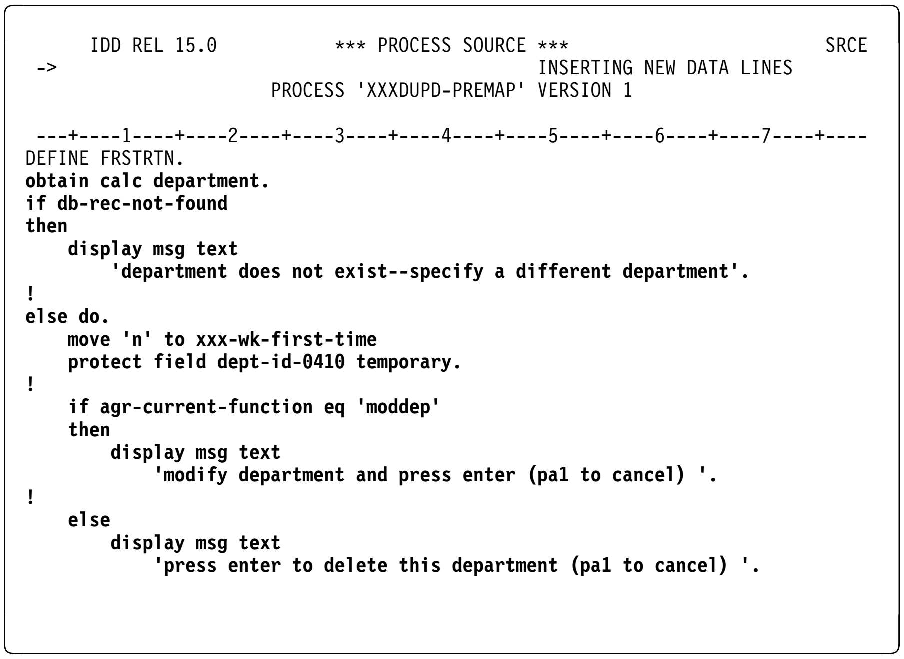
For process module *XXX*DUPD-ENTER, you open up new lines at the end of the Process Source screen and enter more source commands. Place the cursor on the final line of text and press the control key that inserts new lines on IDD screens at your site.



```
      IDD REL 15.0           *** PROCESS SOURCE ***                        SRCE
 ->                                             NO DATA LINES CURRENTLY EXIST
                         PROCESS 'XXXDUPD-PREMAP' VERSION 1

 ---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----
ready usage-mode update
!
if agr-current-response ne spaces
and no fields changed
and xxx-wk-first-time eq 'y'
then
    execute next function.
!
if xxx-wk-first-time eq 'y'
then
    call frstrtn.
!
if agr-current-function eq 'moddep'
then
    call modrtn.
else
    call delrtn.
!
define frstrtn.
```

[PF4]

Add source statements to the end of the process. Notice that the final line from the previous page of the Process Source screen is displayed as the first line of this screen.



```
      IDD REL 15.0           *** PROCESS SOURCE ***                        SRCE
 ->                                             INSERTING NEW DATA LINES
                         PROCESS 'XXXDUPD-PREMAP' VERSION 1

 ---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----
DEFINE FRSTRTN.
obtain calc department.
if db-rec-not-found
then
    display msg text
        'department does not exist--specify a different department'.
!
else do.
    move 'n' to xxx-wk-first-time
    protect field dept-id-0410 temporary.
!
    if agr-current-function eq 'moddep'
    then
        display msg text
            'modify department and press enter (pa1 to cancel) '.
!
    else
        display msg text
            'press enter to delete this department (pa1 to cancel) '.
```

[PF4]

Place the cursor on the final source line and use [PF4] to insert another page of source statements.

```
      IDD REL 15.0           *** PROCESS SOURCE ***                        SRCE
 ->                                             INSERTING NEW DATA LINES
                         PROCESS 'XXXDUPD-PREMAP' VERSION 1

 ---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----
            'PRESS ENTER TO DELETE THIS DEPARTMENT (PA2 TO CANCEL)'.
end
!
define modrtn.
modify department.
!
initialize (department).
!
move 'y' to xxx-wk-first-time
display msg text
        'department modified--specify another department to modify'.
!
define delrtn.
erase department allowing ('0230').
!
move 'y' to xxx-wk-first-time
!
if error-status eq '0230'
then
```

[PF4]

Entering the final page of source statements: Type source statements on the screen. Don't forget to place the periods outside of the single quote for DISPLAY commands.

```
      IDD REL 15.0           *** PROCESS SOURCE ***                        SRCE
 ->                                             INSERTING NEW DATA LINES
                         PROCESS 'XXXDUPD-PREMAP' VERSION 1

 ---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----
THEN
end
    display msg text
        'cannot delete this department--specify another department'.
!
else do.
    move 'y' to xxx-wk-first-time
    initialize (department).
!
    display msg text
        department deleted--specify another department to delete'.
end.
```

```
                                                        [Enter]
```

After you press [Enter], IDD adds the process module source commands to the data dictionary, and then redisplays the Process Source screen with a message like:

```
PROCESS 'XXXDUPD-ENTER' VERSION 1 MODIFIED
```

It is a good idea at this stage to inspect the process module for any syntax errors (such as omitted single quotation marks or periods). You can page the Process Source screen forward and backward:

- **Press [PF8]** (default) to page forward.
- **Press [PF7]** (default) to page backward.

If you notice any errors, you can type over the errors to correct them. After you correct all errors, you can press [Enter] again.

After you finish defining process module *XXX*DUPD-ENTER, you can define process *XXX*DUPD-PA2.

15.3.3 Step 3: Define process module XXXDUPD-PA2

Process module *XXX*DUPD-PA2 is the final process module that you will create for dialog *XXX*DUPD. Sample commands for process module *XXX*DUPD-PA2 are shown below. Statements in this process module allow the end user to cancel the current modification or deletion operation.

```
MOVE 'Y' TO XXX-WK-FIRST-TIME.                                       -┐
INITIALIZE (DEPARTMENT).                                             -┘  1


IF AGR-CURRENT-FUNCTION EQ 'MODDEP'                                  -┐
THEN                                                                  │  2
    DISPLAY MSG TEXT                                                  │
        'MODIFICATION CANCELLED--SPECIFY A DEPARTMENT TO MODIFY'.    -┘

ELSE                                                                 -┐
    DISPLAY MSG TEXT                                                  │  3
        'DELETION CANCELLED--SPECIFY A DEPARTMENT TO DELETE'.        -┘
```

1 These statements set the flag in *XXX*-WK-FIRST-TIME to Y and then initialize the dialog buffers for DEPARTMENT data.

2 For the MODDEP function, the dialog's map is redisplayed with this MODIFICATION CANCELLED message.

3 For the DELDEP function, the dialog's map is redisplayed with this DELETION CANCELLED message.

To define process module *XXX*DUPD-PA2, you use the Process Entity and Process Source screens:

```
-> proc
                                                          [Enter]
```

Enter the screen identifier (PROC) in the command area.

Enter the indicated specifications on the Process Entity screen. Don't forget to deselect the DISPLAY action.

```
     IDD REL 15.0                *** PROCESS ENTITY ***                  PROC
 ->
                                 DICT=DEMO

   DISPLAY      PROCESS NAME....: xxxdupd-pa2
 _ MODIFY
 x ADD         VERSION NUMBER..: 1       _ HIGHEST    _ NEXT HIGHEST
 _ DELETE                                _ LOWEST     _ NEXT LOWEST

               DESCRIPTION.....: cancel mod/del department operation
```

```
                                                          [Enter]
```

After specifying basic information for process module *XXX*DUPD-PA2, you can proceed to add source commands on the Process Source screen:

```
-> srcs
                                                          [Enter]
```

SRCE identifies the Process Source screen.

The Process Source screen is displayed. Enter a full screen of source statements. Enter keywords, periods, and quotes as shown. Don't type characters beyond column 72.

```
     IDD REL 15.0           *** PROCESS SOURCE ***                       SRCE
 ->                                            NO DATA LINES CURRENTLY EXIST
                      PROCESS 'XXXDUPD-PREMAP' VERSION 1

 ---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----
move 'y' to xxx-first-time
initialize (department).
!
if agr-current-function eq 'moddep'
then
    display msg text
        'modification cancelled--specify a department to modify'.
else
    display msg text
        'deletion cancelled--specify a department to delete'.
```

```
                                                                    [Enter]
```

After you press [Enter], IDD adds the process module source commands to the data dictionary and then redisplays the Process Source screen with a message like:

```
PROCESS 'XXXDUPD-PA2' VERSION 1 MODIFIED
```

If you notice any errors on the redisplayed screen, you can type over the errors to correct them, and then press [Enter] again.

After you finish defining process module *XXX*DUPD-PA2, you can exit from the IDD menu facility. In this sample session, you will transfer from IDD to ADSC in order to associate the process modules you defined above with dialog *XXX*DUPD. To do this, you use the SWITCH command:

```
-> switch adsct
                                                                    [Enter]
```

15.4 Completing dialog XXXDUPD using ADSC

Now that you have defined process modules for dialog *XXX*DUPD, you can complete the dialog. To do this, you will perform the following steps:

1. Retrieve dialog *XXX*DUPD.
2. Add a subschema to the dialog.
3. Define dialog options for use during development.
4. Add a work record to the dialog.
5. Add premap and response processes to the dialog.
6. Recompile the dialog load module.

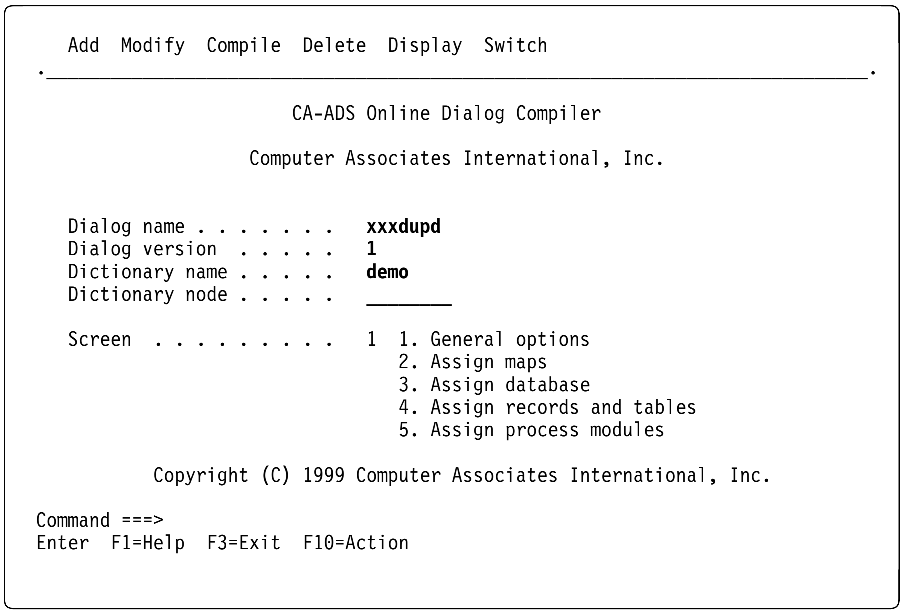
Each of these steps is presented below.

15.4.1 Step 1: Retrieve dialog XXXDUPD

In order to retrieve a dialog, you must be using ADSC.

If you *did not* transfer to ADSC earlier in this chapter when you exited from the IDD menu facility, you need to invoke ADSC. To do this, enter the task code (for example, **ADSCT**) for ADSC. For more information on invoking ADSC, see 9.3.1, “Step 1: Invoke ADSC” on page 9-8.

ADSC begins by displaying the Main Menu screen. To retrieve a dialog, you identify the dialog on a blank **Main Menu** screen:

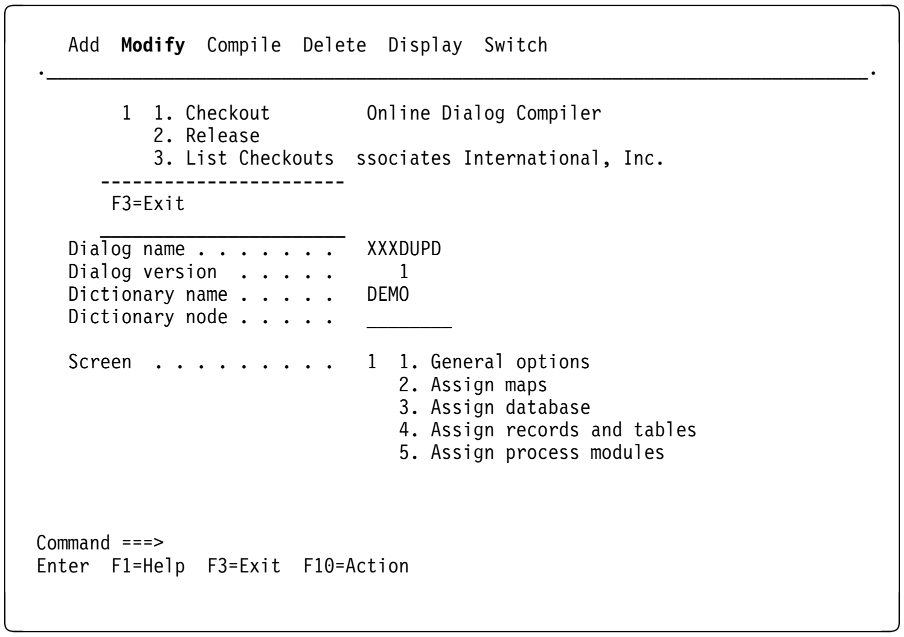


```
   Add  Modify  Compile  Delete  Display  Switch
._____________________________________________________________________________.

                        CA-ADS Online Dialog Compiler

                    Computer Associates International, Inc.


   Dialog name . . . . . . .   xxxdupd
   Dialog version  . . . . .   1
   Dictionary name . . . . .   demo
   Dictionary node . . . . .   ________

   Screen  . . . . . . . . .   1  1. General options
                                  2. Assign maps
                                  3. Assign database
                                  4. Assign records and tables
                                  5. Assign process modules

           Copyright (C) 1999 Computer Associates International, Inc.

Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

[PF10]

To check the application out for modification, position the cursor on the **Modify** item on the action bar and press [Enter]. You can position the cursor on **Modify** by:

- Pressing [PF10] to move to the action bar and then tabbing to **Modify** and pressing [Enter]
- Tabbing to **Modify** and pressing [Enter]
- Typing **modify** on the command line and pressing [Enter]



```
    Add  Modify  Compile  Delete  Display  Switch
 .______________________________________________________________________________.

         1  1. Checkout        Online Dialog Compiler
            2. Release
            3. List Checkouts  ssociates International, Inc.
       -----------------------
        F3=Exit
       _______________________
    Dialog name . . . . . . .   XXXDUPD
    Dialog version  . . . . .      1
    Dictionary name . . . . .   DEMO
    Dictionary node . . . . .   ________

    Screen  . . . . . . . . .   1  1. General options
                                   2. Assign maps
                                   3. Assign database
                                   4. Assign records and tables
                                   5. Assign process modules



 Command ===>
 Enter  F1=Help  F3=Exit  F10=Action
```

[Enter]

Press [Enter] to check the application out.

Note: If the dialog has been checked out to another developer and has not been released by that developer, you will not be able to check it out.

After you press [Enter], ADSC displays dialog *XXX*DUPD on the Main Menu screen, along with the following messages:

```
NO SCHEMA/SUBSCHEMA - NO DATABASE CALLS ALLOWED
NO ERRORS DETECTED
```

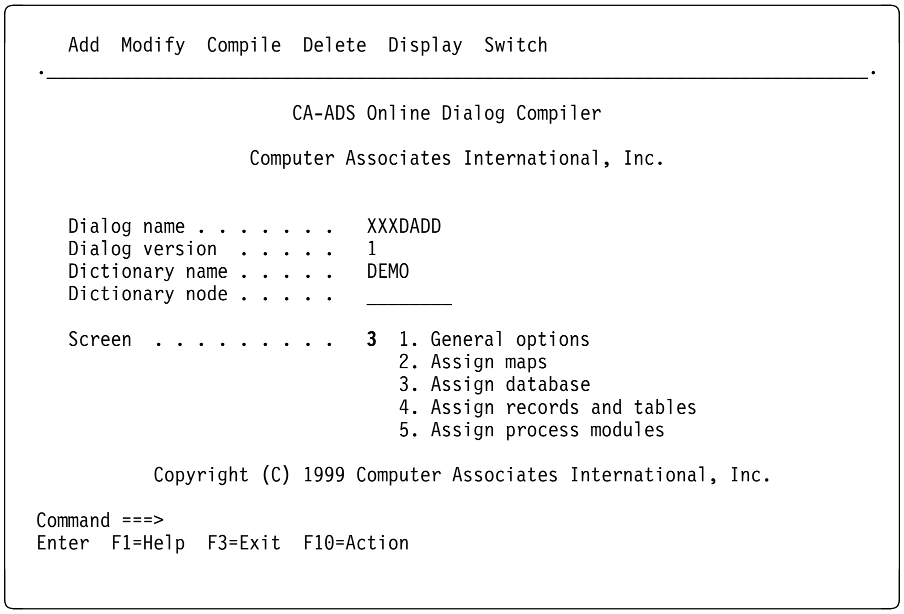
The **NO ERRORS DETECTED** message is the message you should look for after retrieving a dialog definition. After you successfully retrieve *XXX*DUPD, you can associate a subschema with the dialog.

15.4.2 Step 2: Add a subschema

In this step, you will associate a subschema with dialog *XXX*DUPD so that process modules for the dialog can retrieve, modify, and delete departments in the database at run time.

You use the **Database Specifications** screen to associate a subschema with a dialog. For example:

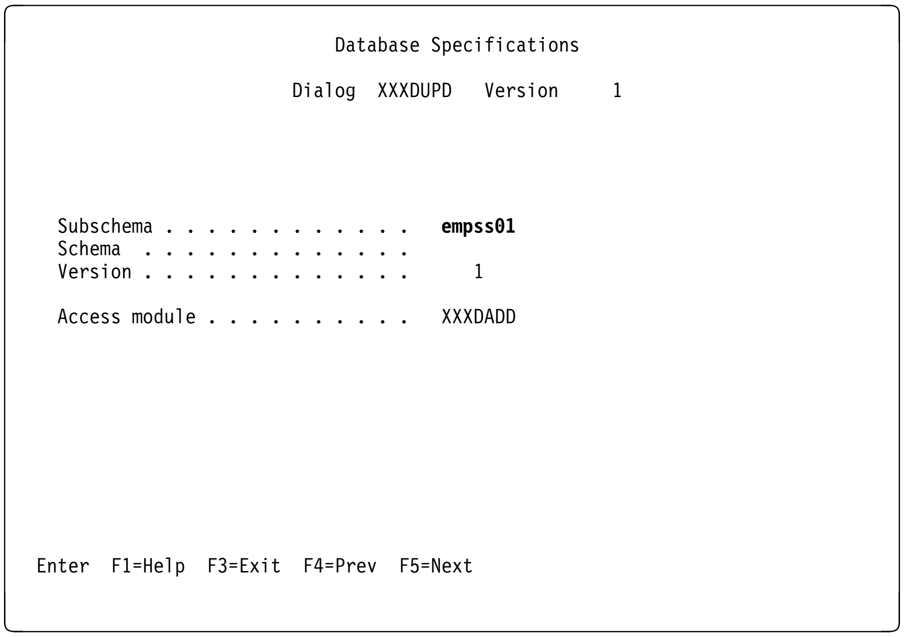
Accessing the Database Specifications screen: To access the Database Specifications screen, you enter **3** at the **Screen** prompt on the Main Menu.



```
   Add  Modify  Compile  Delete  Display  Switch
._____________________________________________________________________________.

                      CA-ADS Online Dialog Compiler

                  Computer Associates International, Inc.


   Dialog name . . . . . . .   XXXDADD
   Dialog version  . . . . .   1
   Dictionary name . . . . .   DEMO
   Dictionary node . . . . .   ________

   Screen  . . . . . . . . .   3  1. General options
                                  2. Assign maps
                                  3. Assign database
                                  4. Assign records and tables
                                  5. Assign process modules

           Copyright (C) 1999 Computer Associates International, Inc.

Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

`[Enter]`

Sample Database Specifications screen: The Database Specifications screen is displayed.


```
                          Database Specifications

                     Dialog  XXXDUPD   Version     1




   Subschema . . . . . . . . . . . .   empss01
   Schema  . . . . . . . . . . . . .
   Version . . . . . . . . . . . . .      1

   Access module . . . . . . . . . .   XXXDADD









 Enter  F1=Help  F3=Exit  F4=Prev  F5=Next
```

For more information on subschemas, see 12.4.3, "Step 3: Add a subschema" on page 12-24.

When you press the [Enter] key, ADSC associates the named subschema with the dialog if there are no errors, and then redisplays the screen with the following message:

```
NO ERRORS DETECTED
```

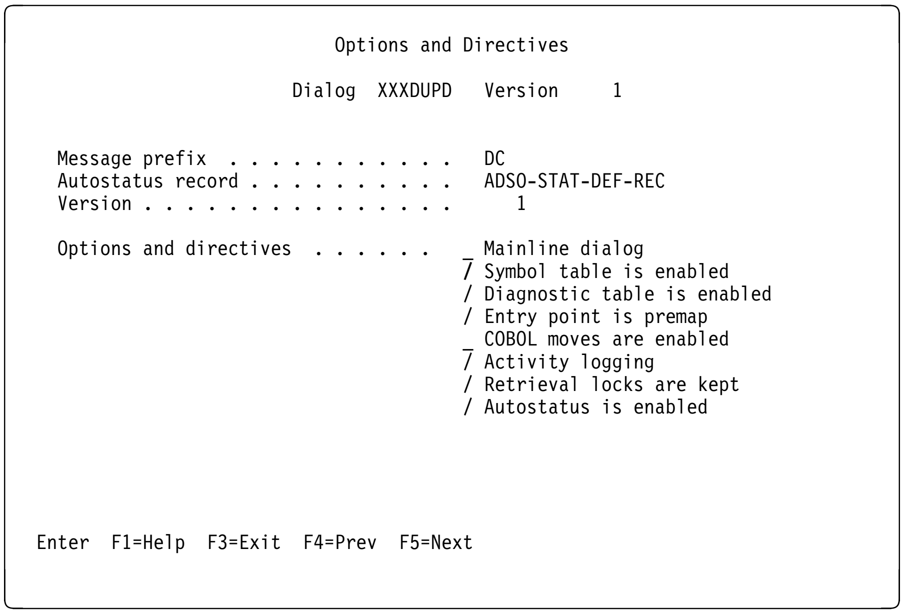
If a different message is displayed, read the message to determine the problem. If you did not specify a schema name, ask others at your site whether a schema name is required. After you type new values, press [Enter] again.

After you successfully associate a subschema with the dialog, you can proceed to add work record *XXX*-WK-RECORD to the dialog.

15.4.3 Step 3: Define dialog options

You use the **Options and Directives** screen to specify options to help you develop and debug dialogs. In this chapter, you use the Options and Directives screen to add a **symbol table** and a **diagnostic table** to dialog *XXX*DUPD. These tables are often useful when debugging dialogs.

You can display and use the Options and Directives screen as shown: The diagnostic table is enabled by default. Enable the symbol table.


```
                            Options and Directives

                        Dialog  XXXDUPD   Version     1


   Message prefix  . . . . . . . . . . .   DC
   Autostatus record . . . . . . . . . .   ADSO-STAT-DEF-REC
   Version . . . . . . . . . . . . . . .      1

   Options and directives  . . . . . .   _ Mainline dialog
                                         / Symbol table is enabled
                                         / Diagnostic table is enabled
                                         / Entry point is premap
                                         _ COBOL moves are enabled
                                         / Activity logging
                                         / Retrieval locks are kept
                                         / Autostatus is enabled




 Enter  F1=Help  F3=Exit  F4=Prev  F5=Next
```

`[Enter]`

After you press [Enter], the Options and Directives screen displays the following message to inform you that your specifications contain no errors:

```
INPUT HAS BEEN SUCCESSFULLY PROCESSED
```

For more information on the Options and Directives screen, see 12.4.2, “Step 2: Specify dialog options” on page 12-23.

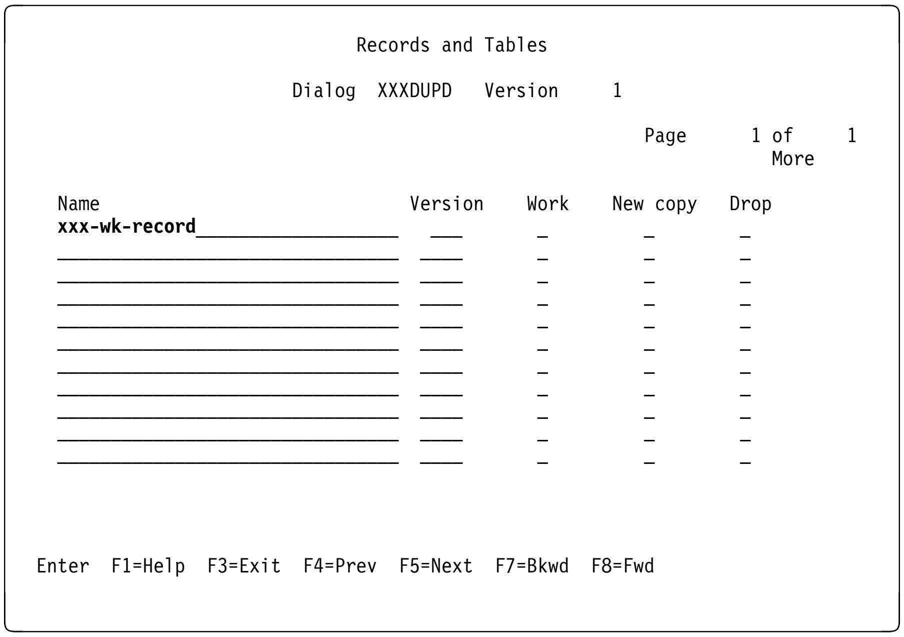
15.4.4 Step 4: Add a work record

Process modules that you defined for dialog *XXX*DUPD use element *XXX*-WK-FIRST-TIME. To enable the process modules to use the element at run time, you add to the dialog the work record (*XXX*-WK-RECORD) that contains the element.

Some work records define data to be displayed on the dialog's map. In this case, the work record is automatically added to the dialog when the map is added.

Work record *XXX*-WK-RECORD *does not* define data for display on the dialog's map. Therefore, you need to explicitly add the record to the dialog. To do this, you use the ADSC **Records and Tables screen**.

You access and use the Records and Tables screen as shown:



```
                          Records and Tables

                     Dialog  XXXDUPD   Version     1

                                                Page      1 of     1
                                                            More

   Name                             Version    Work    New copy   Drop
   xxx-wk-record___________________   ___       _         _        _
   ________________________________  ____       _         _        _
   ________________________________  ____       _         _        _
   ________________________________  ____       _         _        _
   ________________________________  ____       _         _        _
   ________________________________  ____       _         _        _
   ________________________________  ____       _         _        _
   ________________________________  ____       _         _        _
   ________________________________  ____       _         _        _
   ________________________________  ____       _         _        _
   ________________________________  ____       _         _        _



 Enter  F1=Help  F3=Exit  F4=Prev  F5=Next  F7=Bkwd  F8=Fwd
```

When you press [Enter], ADSC associates the named work record with the dialog if there are no errors, and then redisplays the Records and Tables screen with a message:

- **Records and Tables PROCESSED SUCCESSFULLY** is displayed when you successfully associate a work record with the dialog.
- **A different message** is displayed if ADSC detects any errors.

  In this case, read the message to determine the problem. If the work record was provided to you by other developers at your site, find out if you need to supply a version number for the record under the **Version** prompt. You can type over any mistakes and press [Enter] again.

After you successfully associate work record *XXX*-WK-RECORD with dialog *XXX*DUPD, you can proceed to add process modules to the dialog.

15.4.5 Step 5: Add premap and response processes

You use the ADSC **Process Modules** screen to associate premap and response processes with a dialog

Earlier in this chapter, you defined a premap process, *XXX*DUPD-PREMAP, and two response processes for dialog *XXX*DUPD: *XXX*DUPD-ENTER and *XXX*DUPD-PA2.

The premap process will execute before the map is displayed to the end user.

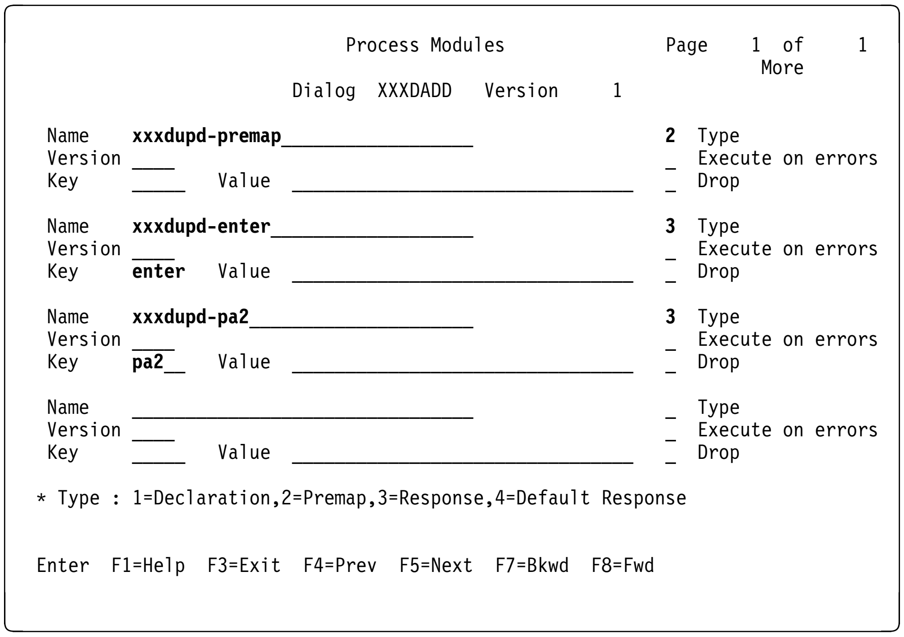
When the end user presses a control key to input data on the dialog's map, one and only one of the response processes can be executed. To make it easy for the end user to select the appropriate response process, you will associate each response process with its own control key.

As an application developer, you associate control keys with response processes when you add the processes to the dialog:

- When you add *XXX*DUPD-ENTER, you will specify that this response process is invoked when the end user presses the **[Enter] key**.
- When you add *XXX*DUPD-PA2, you will specify that this response process is invoked when the end user presses [PA2]:

You access the Process Modules screen by entering **5** at the **Screen** prompt on the Main Menu and pressing [Enter].

Process Modules screen: The Process Modules screen is displayed. You type the name of the process module after the **Name** prompt:


```
                          Process Modules              Page    1  of     1
                                                                More
                     Dialog  XXXDADD   Version     1

  Name    xxxdupd-premap__________________                  2  Type
  Version ____                                              _  Execute on errors
  Key     _____   Value  ________________________________   _  Drop

  Name    xxxdupd-enter___________________                  3  Type
  Version ____                                              _  Execute on errors
  Key     enter   Value  ________________________________   _  Drop

  Name    xxxdupd-pa2_____________________                  3  Type
  Version ____                                              _  Execute on errors
  Key     pa2__   Value  ________________________________   _  Drop

  Name    ________________________________                  _  Type
  Version ____                                              _  Execute on errors
  Key     _____   Value  ________________________________   _  Drop

 * Type : 1=Declaration,2=Premap,3=Response,4=Default Response


 Enter  F1=Help  F3=Exit  F4=Prev  F5=Next  F7=Bkwd  F8=Fwd
```

[Enter]

When you press [Enter], ADSC compiles the process module source code and adds the compiled representation of each module to the dialog if there are no errors. ADSC redisplays the Process Modules screen with a message:

- **PROCESS HAS COMPILED SUCCESSFULLY** is displayed when the compiled process module was successfully added to the dialog as a premap process.
- **TO SEE ERRORS SELECT DISPLAY OR PRINT** is displayed when ADSC cannot successfully compile the process module because of errors.

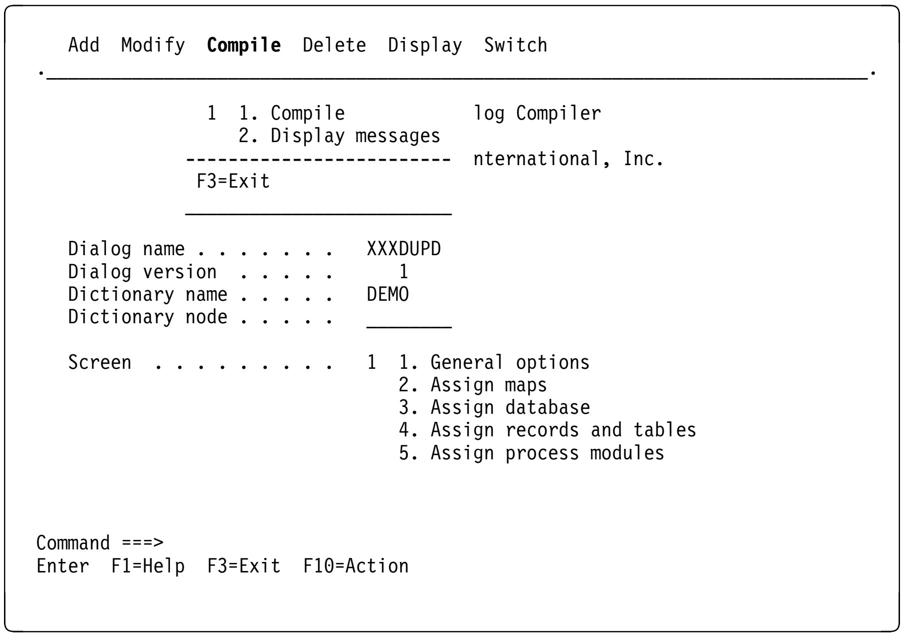
  In this case, you display and correct errors in the process module, as discussed in 12.4.6, "Correct errors in process modules" on page 12-34.

After the process modules have compiled successfully, you can recompile the dialog.

15.4.6 Step 6: Recompile the dialog

In this chapter, you completed dialog *XXX*DUPD by adding a subschema, dialog options, a work record, a premap process, and two response processes.

To update the load module for dialog *XXX*DUPD, recompile the dialog by selecting the **Compile** activity:



```
    Add  Modify  Compile  Delete  Display  Switch
 .______________________________________________________________________________.

                 1  1. Compile           log Compiler
                    2. Display messages
               -------------------------  nternational, Inc.
                F3=Exit
               _________________________

    Dialog name . . . . . . .   XXXDUPD
    Dialog version  . . . . .      1
    Dictionary name . . . . .   DEMO
    Dictionary node . . . . .   ________

    Screen  . . . . . . . . .   1  1. General options
                                   2. Assign maps
                                   3. Assign database
                                   4. Assign records and tables
                                   5. Assign process modules



 Command ===>
 Enter  F1=Help  F3=Exit  F10=Action
```

`[Enter]`

When you press [Enter], ADSC attempts to recompile the dialog. ADSC then displays a message:

- **DIALOG HAS BEEN MODIFIED** is displayed on the Dialog Definition screen when the dialog was successfully recompiled.
- **A different message** is displayed if an error in the dialog definition prevents ADSC from recompiling the dialog.

  In this case, correct the errors and then recompile the dialog as described above.

After you successfully recompile dialog *XXX*DUPD, you can exit from ADSC by using [PF3].

After you exit from ADSC, you can execute the application.

15.5 Executing the application

Now that you have enhanced dialog *XXX*DUPD, the sample Department application is complete. You can execute the application to test out all application features and see how end users would use them. You will:

1. Invoke the Department application.
2. Display function MODDEP, which executes dialog *XXX*DUPD.
3. Try out process logic that modifies sample department records.
4. Display function DELDEP, which also executes dialog *XXX*DUPD, and try out process logic that deletes sample department records.

When testing functions MODDEP and DELDEP, you can modify and delete the sample departments that you added when you tested function ADDDEP in Chapter 12, "Adding Process Logic to a Dialog" on page 12-1. When you modify departments in this chapter, it is a good idea to **write down** the changes that you make.

To invoke the Department application from IDMS-DC/UCF, you enter the task code (*XXX*DEPT) for the application. For example, when using IDMS-DC, you invoke the Department application as shown:

```
ENTER NEXT TASK CODE:
xxxdept

                                                      [Enter]
```

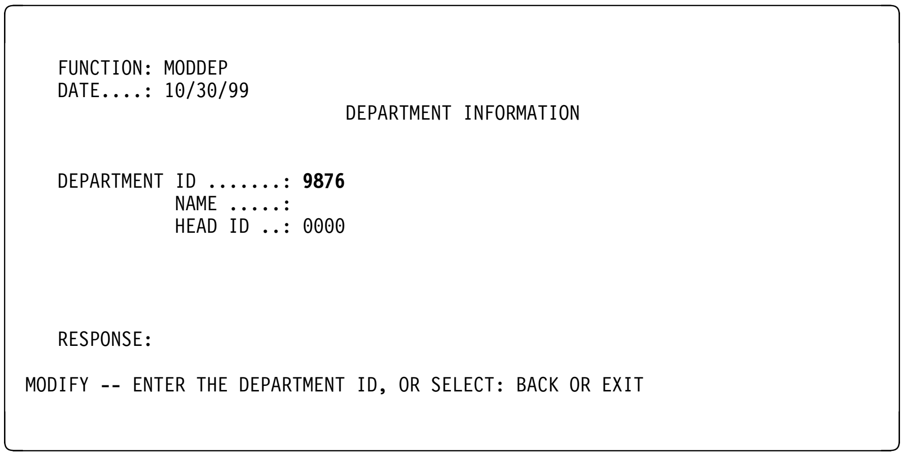
For more information on invoking the application, see 9.4, "Instructions for executing the application" on page 9-17.

You can display the MODDEP function and modify a department, as shown below for a sample department named TEST DEPARTMENT:

```
x MOD
                                                      [Enter]
```

Select the MOD response to display MODDEP.

The MODDEP function screen is displayed. Specify the department to be modified.



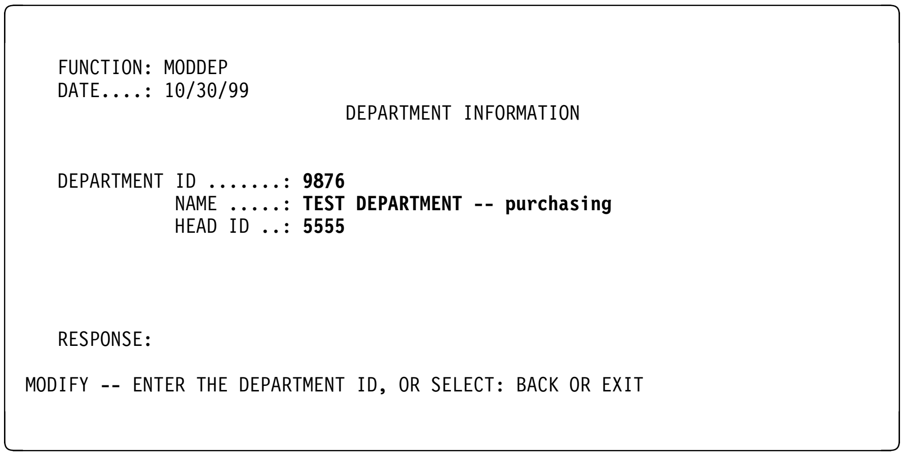
```
   FUNCTION: MODDEP
   DATE....: 10/30/99
                            DEPARTMENT INFORMATION


   DEPARTMENT ID .......: 9876
              NAME .....:
              HEAD ID ..: 0000




   RESPONSE:

MODIFY -- ENTER THE DEPARTMENT ID, OR SELECT: BACK OR EXIT
```

[Enter]

You defined the message in premap process *XXX*DUPD-PREMAP.

Modify the displayed information by typing over and erasing old values.

Remember that to promote data integrity, you protected the DEPARTMENT ID variable field (which contains a department's unique ID number) from user updates.

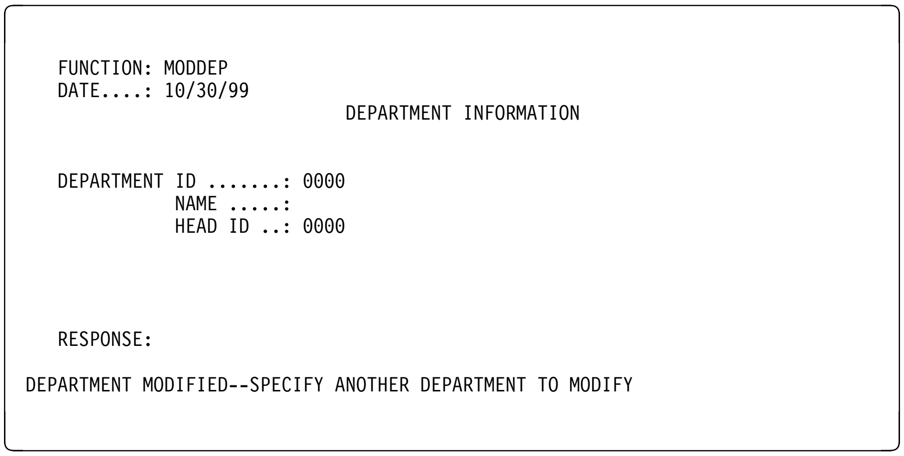


```
   FUNCTION: MODDEP
   DATE....: 10/30/99
                            DEPARTMENT INFORMATION


   DEPARTMENT ID .......: 9876
              NAME .....: TEST DEPARTMENT -- purchasing
              HEAD ID ..: 5555




   RESPONSE:

MODIFY -- ENTER THE DEPARTMENT ID, OR SELECT: BACK OR EXIT
```

[Enter]

The department is modified.

Notice the message you defined in response process *XXX*DUPD-ENTER for display when department data has been modified.



```
   FUNCTION: MODDEP
   DATE....: 10/30/99
                              DEPARTMENT INFORMATION


   DEPARTMENT ID .......: 0000
              NAME .....:
              HEAD ID ..: 0000




   RESPONSE:

DEPARTMENT MODIFIED--SPECIFY ANOTHER DEPARTMENT TO MODIFY
```

The process logic that you defined for dialog *XXX*DUPD allows you to modify department records in two steps:

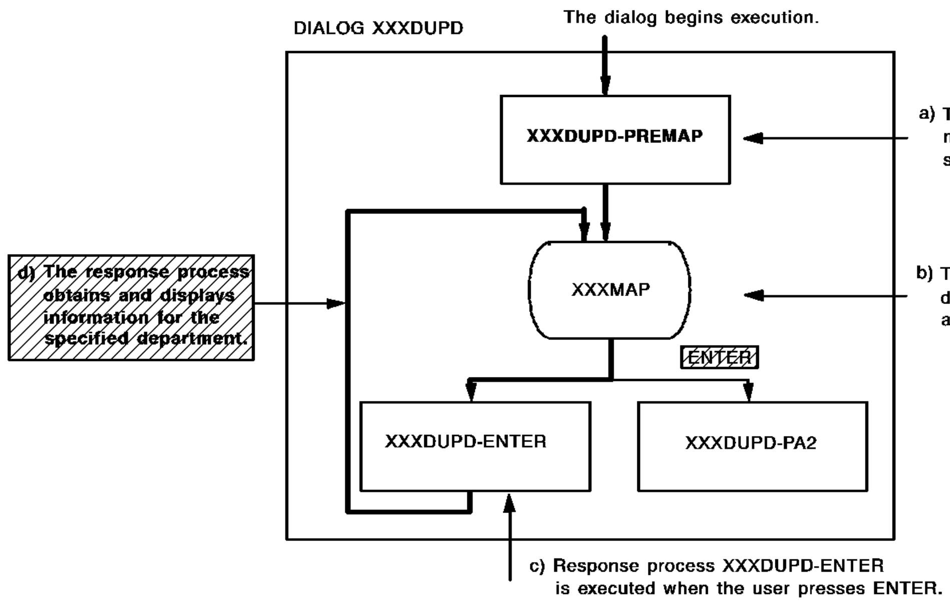
1. The first time you press [Enter], response process *XXX*DUPD-ENTER retrieves the specified department from the database.
2. The second time you press [Enter], the same response process modifies information for the department in the database.

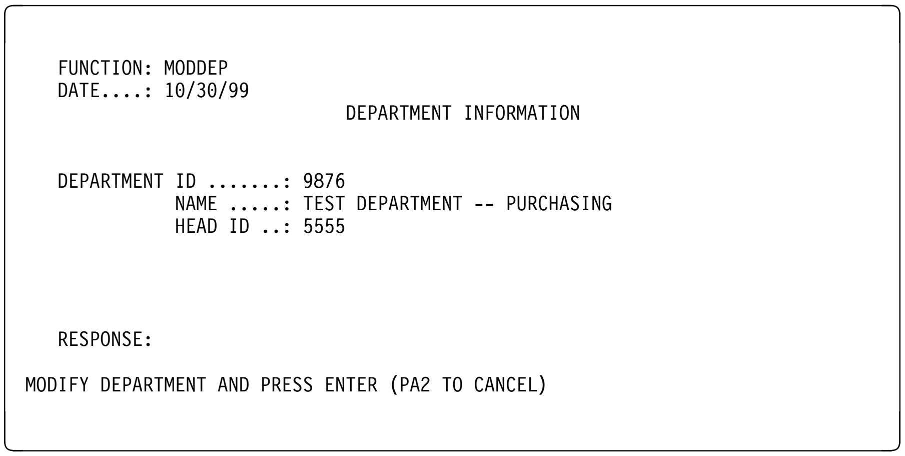
The following diagram shows how components for dialog *XXX*DUPD are executed at run time when you use the dialog to modify a department record. Response process XXXDUPD-ENTER executes twice when you use dialog XXXDUPD to modify a department.

After you modify a department, you can verify that your modifications have been updated to the database by specifying the department ID again:

```
DEPARTMENT ID.......:  9876
                                                       [Enter]
```

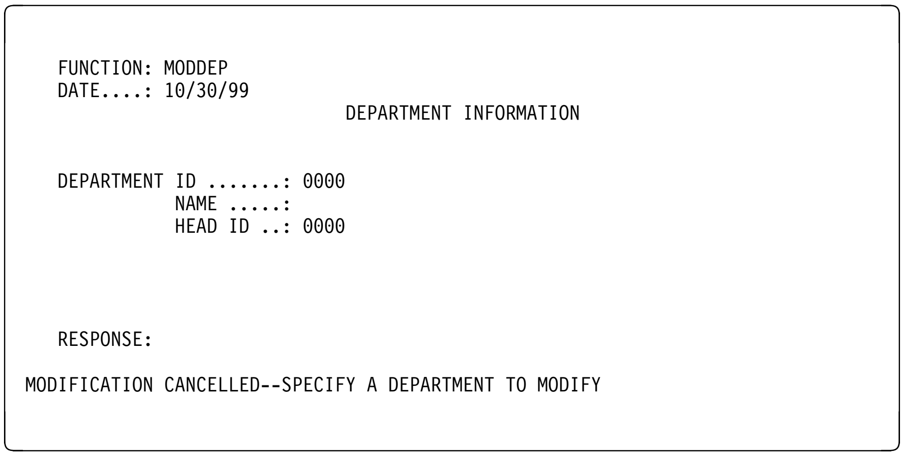
Modified values are retrieved from the database and the modified department is redisplayed.


DIALOG XXXDUPD
The dialog begins execution.
XXXDUPD-PREMAP
a) This process displays the map prompting the user to specify a department.
XXXMAP
b) The user identifies the department to be modified and presses ENTER.
d) The response process obtains and displays information for the specified department.
ENTER
XXXDUPD-ENTER
XXXDUPD-PA2
c) Response process XXXDUPD-ENTER is executed when the user presses ENTER.



```
   FUNCTION: MODDEP
   DATE....: 10/30/99
                              DEPARTMENT INFORMATION


   DEPARTMENT ID .......: 9876
              NAME .....: TEST DEPARTMENT -- PURCHASING
              HEAD ID ..: 5555




   RESPONSE:

MODIFY DEPARTMENT AND PRESS ENTER (PA2 TO CANCEL)
```

Instead of modifying this department again, you can try pressing [PA2] to see how process module *XXX*DUPD-PA2 allows you to cancel a modification operation:



```
   FUNCTION: MODDEP
   DATE....: 10/30/99
                              DEPARTMENT INFORMATION


   DEPARTMENT ID .......: 0000
              NAME .....:
              HEAD ID ..: 0000




   RESPONSE:

MODIFICATION CANCELLED--SPECIFY A DEPARTMENT TO MODIFY
```

You defined this message in response process *XXX*DUPD-PA2 to confirm cancellation of a modification operation.

When you press [PA2], response process *XXX*DUPD-PA2 is executed. This response process cancels your current operation without modifying the database. The following diagram shows how components for dialog *XXX*DUPD are executed when you press [PA2] to cancel a modification operation. XXXDUPD-PA2 and cancel a modification operation.

While executing the Department application, you also can test out how dialog *XXX*DUPD handles deletion operations. To do this:

1. Transfer from MODDEP to DELDEP.
2. Then, specify the ID number of a department to be deleted.

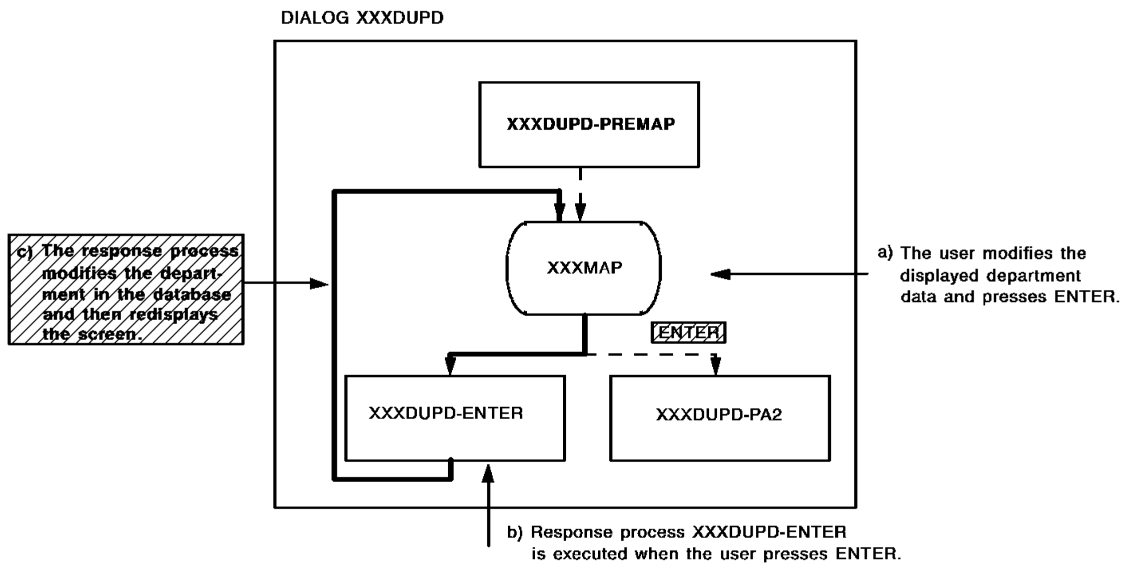
Deleting a sample department: Function DELDEP allows you to delete existing departments from the database. You should be careful, when you test this function, that you delete only your own sample departments from the database.

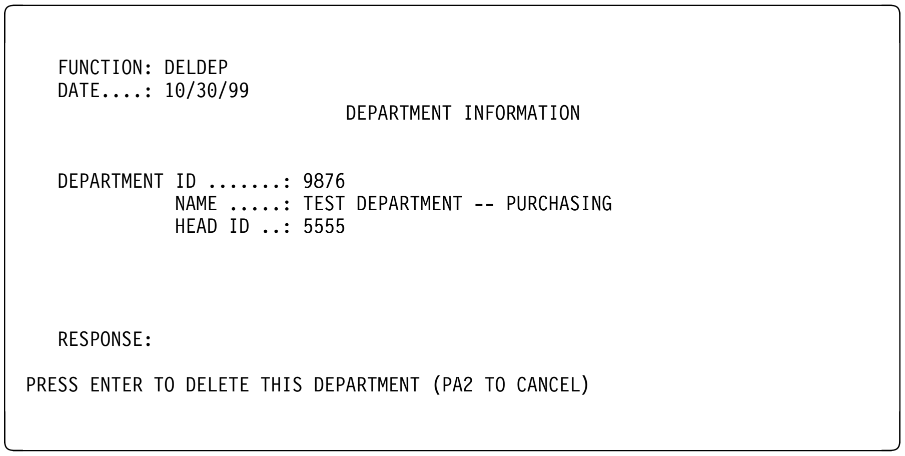
You can use the DELDEP function to delete a department.

Specify the department to be deleted by entering the ID number for an existing department.

```
DEPARTMENT ID........: 9876
                                                       [Enter]
```

The department record is display for your verification.


DIALOG XXXDUPD
XXXDUPD-PREMAP
XXXMAP
ENTER
XXXDUPD-ENTER
XXXDUPD-PA2
a) The user modifies the displayed department data and presses ENTER.
b) Response process XXXDUPD-ENTER is executed when the user presses ENTER.
c) The response process modifies the department in the database and then redisplays the screen.



```
   FUNCTION: DELDEP
   DATE....: 10/30/99
                              DEPARTMENT INFORMATION


   DEPARTMENT ID .......: 9876
              NAME .....: TEST DEPARTMENT -- PURCHASING
              HEAD ID ..: 5555




   RESPONSE:

PRESS ENTER TO DELETE THIS DEPARTMENT (PA2 TO CANCEL)
```

`[Enter]`

When you press [Enter], the screen is redisplayed with the message you defined in response process *XXX*DUPD-ENTER:

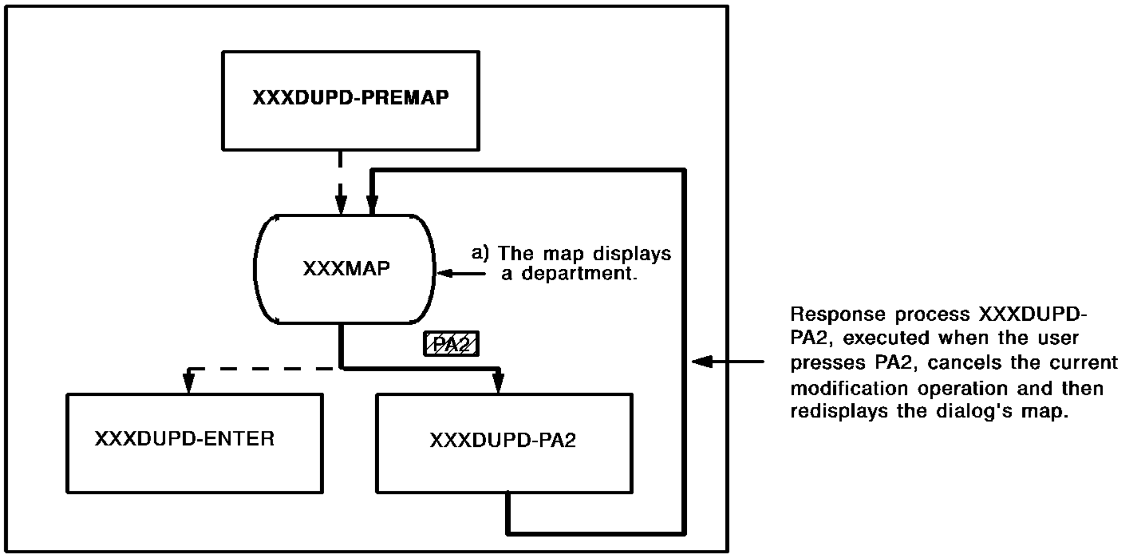
```
DEPARTMENT DELETED--SPECIFY ANOTHER DEPARTMENT TO DELETE
```

The process logic that you defined for dialog *XXX*DUPD allows you to delete a department record in two steps:

1. The first time you press [Enter], process module *XXX*DUPD-ENTER retrieves and displays the specified department.
2. The second time you press [Enter], process module *XXX*DUPD-ENTER deletes the department from the database.

You can verify that process module *XXX*DUPD-ENTER actually deleted the department from the database by specifying the department ID number again:

DIALOG XXXDUPD


XXXDUPD-PREMAP
XXXMAP
a) The map displays a department.
PA2
XXXDUPD-ENTER
XXXDUPD-PA2
Response process XXXDUPD-PA2, executed when the user presses PA2, cancels the current modification operation and then redisplays the dialog's map.

```
 DEPARTMENT ID........: 9876
                                                 [Enter]
```

When you press [Enter], the screen is redisplayed with the message you defined in response process *XXX*DUPD-ENTER:

```
DEPARTMENT DOES NOT EXIST--PRESS ENTER TO SPECIFY ANOTHER
```

While testing dialog *XXX*DUPD, you also can verify that [PA2] allows an end user to cancel a deletion operation. To do this, you specify an existing department, as shown:

```
DEPARTMENT ID.........: 9876

                                                  [Enter]
```

Then, when dialog *XXX*DUPD displays the department record, you press [PA2].

Before you exit from the application, you can use the DELDEP function to delete all sample departments that you've added to the database.

While you are executing this production version of the Department application, you also can test out all other capabilities of the sample Department application to see how the application allows you to perform all operations needed to add, modify, and delete departments in the database.

When you are ready to exit from the Department application, you can select the EXIT response.

15.6 Summary

The sample Department application is now fully developed. The steps you performed while developing the Department application throughout this manual could have been performed in a different order, depending on the preferences and requirements of the site. For example, you could have added process logic to dialogs before associating the dialogs with ADSA application functions.

In Part II of this manual, "Developing the Prototype," you began developing the Department application based on a structure diagram for the application. You **created a prototype** of the application by defining components for the application:

1. **You defined the executable application structure** in Chapter 7, “Defining an Application Structure Using ADSA” on page 7-1. The structure consists of:

   - **Functions** (for example, MODDEP) that represent units of work (such as dialogs) to be performed by the application
   - **Responses** (for example, EXIT) that define run-time paths between functions in the application
   - **A task code** (*XXX*DEPT) that defines an entry point for the application

2. **You defined a map** in Chapter 8, “Defining a Screen Display Using MAPC” on page 8-1. The map allows dialogs in the application to display department information.

3. **You defined skeleton versions of dialogs** in Chapter 9, “Defining Dialogs Using ADSC” on page 9-1. At run time, these dialogs are executed when control passes to the application functions that invoke them.

Based on tests of the completed prototype, you **modified the prototype** to make the application easier to use. You changed flow of control and screen displays without having to alter any code.

After the prototype application was approved, you **developed a fully functional application** from the prototype in Part III of this manual, "Enhancing the Application Definition." You added process logic and a work record to the application's skeleton dialogs:

1. **You defined process modules for dialog *XXX*DADD** in Chapter 12, “Adding Process Logic to a Dialog” on page 12-1 and Chapter 13, “Modifying Process Logic in a Dialog” on page 13-1. These processes allow end users to add new departments to the database.

2. **You defined a work record** in Chapter 14, “Defining Work Records Using IDD” on page 14-1. The record stores a value that determines the processing performed by dialog *XXX*DUPD at run time.

3. **You defined process modules for dialog *XXX*DUPD** in this chapter. These processes allow end users to modify and delete departments in the database.

After completing the Department application, you **executed the completed application** to test its full capabilities.

Even though you have tested the Department application, you still may need to make modifications either now or in the future. For example, consider the following sources of change:

- **Tests and preliminary use of the application** can cause developers, end users, and managers to suggest changes.

  For example, end users of the Department application might prefer blanks to be displayed instead of zeros (0000) in fields where they enter numeric values (for example, the DEPARTMENT ID variable field).

  End users might request that execution of the application be changed from STEP to FAST mode. In FAST mode, the end user can enter values on one screen and transfer to another screen at the same time.

  End users might have difficulty remembering the ID numbers for all departments to be modified or deleted. One solution would be to add a dialog function that alphabetically lists all departments and allows the end user to select a department from the list.

- **Changes in regulations, updated business functions, and other planned and unplanned developments** can require developers to modify an application.

  For example, new contractors might require the collection of additional information for departments.

  Long-term plans for the sample Department application could call for you to add EMPLOYEE and OFFICE functions to the existing application structure.

CA-ADS application development tools can be used at any time to modify individual application components and to add new components to the application. For example, you can add EMPLOYEE and OFFICE functions to the Department application by defining maps and process modules for EMPLOYEE and OFFICE dialogs. When the dialogs are developed, you can incorporate the dialogs into the application structure by using ADSA to add the necessary functions and responses to the application.

Appendix A. Sample Application Components

Components in the sample application: This appendix provides information for all components created for and used by the sample Department application. Components are presented in the order in which you first define them or use them in this manual. Names of components in this table that begin with XXX can be changed when you define the components. For example, you can use your initials instead of XXX.

Component type	Sample name	Characteristics
Application structure	XXXAPPL	Defined in Steps 2 through 8 of Chapter 7. Modified in Steps 2 through 4 of Chapter 10. Contains: • BACK reponse • EXIT response • ADD response • MOD response • DEL response • ADDDEP function • MODDEP function • DELDEP function • DEPTMENU function • XXXDEPT task code
Application responses	BACK	Defined in Steps 4 and 5 of Chapter 7: • Assigned key: CLEAR • Function invoked: POP • Response type: GLOBAL
Application responses	EXIT	Defined in Steps 4 and 5 of Chapter 7: • Assigned key: [PF9] • Function invoked: QUIT • Response type: GLOBAL Modified in Steps 2 and 3 of Chapter 10: • Assigned key: [PF3]
Application responses	ADD	Defined in Steps 4 and 5 of Chapter 7: • Assigned key: [PF4] • Function invoked: ADDDEP • Response type: LOCAL
Application responses	MOD	Defined in Steps 4 and 5 of Chapter 7: • Assigned key: [PF5] • Function invoked: MODDEP • Response type: LOCAL
Application responses	DEL	Defined in Steps 4 and 5 of Chapter 7: • Assigned key: [PF6] • Function invoked: DELDEP • Response type: LOCAL
Application functions	ADDDEP	Defined in Steps 4 and 6 of Chapter 7: • Dialog name: XXXDADD • Function type: 1 (dialog) • Valid responses: BACK EXIT Modified in Steps 2 and 4 of Chapter 10: • Valid responses: MOD BACK EXIT

Component type	**Sample name**	**Characteristics**
Application functions	MODDEP	Defined in Steps 4 and 6 of Chapter 7: • Dialog name: XXXDUPD • Function type: 1 (dialog) • Valid responses: BACK EXIT
Application functions	DELDEP	Defined in Steps 4 and 6 of Chapter 7: • Dialog name: XXXDUPD • Function type: 1 (dialog) • Valid responses: BACK EXIT
Application functions	DEPTMENU	Defined in Steps 4 and 6 of Chapter 7: • Function type.....: 3 (menu) • Valid responses...: ADD MOD DEL EXIT
Task code	XXXDEPT	Defined in Step 7 of Chapter 7: • Assoc. function...: DEPTMENU
Database record	DEPARTMENT	Added to map XXXMAP in Step 3 of Chapter 8; describes department information to be stired in the database.
Map	XXXMAP	Defined in Steps 1 through 9 of Chapter 8. Associated records: • DEPARTMENT • ADSO-APPLICATION-GLOBAL-RECORD Modified in Chapter 11.
Subschema	EMPSS01	Added to dialog XXXDADD in Step 1 in Chapter 12. Added to dialog XXXDUPD in "Completing dialog XXXDUPD using ADSC" in Chapter 15; contains the DEPARTMENT database record and makes it available to dialogs using navigational DML statements.
Dialogs	XXXDADD	Defined in Steps 2 and 3 of Chapter 9: • Associated map....: XXXMAP Updated due to modifications in map XXXMAP in Steps 1 through 3 in Chapter 11. Enhanced in "Adding process modules to dialogs using ADSC" in Chapter 12: • Subschema.........: EMPSS01 • Premap process....: XXXDADD-PREMAP • Response process..: XXXDADD-RESPONSE Modified in "Updating modified process modules in dialogs using ADSC" in Chapter 13.

Component type	Sample name	Characteristics
Dialogs	XXXDUPD	Defined in Step 5 of Chapter 9: • Associated map....: XXXMAP Updated due to modifications in map XXXMAP in "Updating modified maps in dialogs using ADSC" in Chapter 11. Enhanced in "Completing dialog XXXDUPD using ADSC" in Chapter 15: • Subschema.........: EMPSS01 • Premap process....: XXXDUPD-PREMAP • Response processes: XXXDUPD-ENTER XXXDUPD-PA2
Process module	XXXDADD-PREMAP	Defined in "Defining process modules using IDD" in Chapter 12: • Purpose..: Premap process for dialog XXXDADD
Process module	XXXDADD-RESPONSE	Defined in "Defining process modules using IDD" in Chapter 12: • Purpose..: Response process for dialog XXXDADD Modified in "Modifying process modules using IDD" in Chapter 13.
Element	XXX-WK-FIRST-TIME	Defined in Step 2 of Chapter 14. • Purpose..: Stores a single-character status value (Y or N) for process modules in dialog XXXDUPD • Usage ...: DISPLAY • Picture..: X Added to work record XXX-WK-RECORD in Step 3 of Chapter 14.
Work record	XXX-WK-RECORD	Defined in Step 3 of Chapter 14: • Purpose..: Work record for process modules in dialog XXXDUPD • Element..: XXX-WK-FIRST-TIME
Process modules	XXXDUPD-PREMAP	Defined in "Defining process modules using IDD" in Chapter 15: • Purpose..: Premap process for dialog XXXDUPD
Process modules	XXXDUPD-ENTER	Defined in "Defining process modules using IDD" in Chapter 15: • Purpose..: Response process for dialog XXXDUPD

Component type	Sample name	Characteristics
Process modules	XXXDUPD-PA2	Defined in "Defining process modules using IDD" in Chapter 15: • `Purpose..: Response process for dialog XXXDUPD`

Appendix B. Development Tools in the CA-ADS Environment

B.1 Overview

You use the following development tools when you define an CA-ADS application:

- **CA-ADS application compiler (ADSA)** — Used to define the executable structure of an application
- **CA-ADS dialog compiler (ADSC)** — Used to define the dialogs that display and request data at runtime
- **Online mapping facility (MAPC)** — Used to define the screens (maps) displayed by dialogs at runtime
- **Integrated Data Dictionary (IDD) menu facility** — Used to define records and process modules

Types of definitions: The types of definitions created by using ADSA, ADSC, MAPC, and IDD are listed in following table. Following a general discussion of the use of CA-ADS development tools, operations that are commonly performed when using ADSA, ADSC, MAPC, and IDD are presented for each tool. The definitions created by each CA-ADS development tool are listed in this table.

Development tool	Definitions created
CA-ADS application compiler (ADSA)	• Application structures composed of: • Functions • Responses • Task codes
CA-ADS dialog compiler (ADSC)	• Dialogs
Online mapping facility (MAPC)	• Maps
IDD menu facility	• Data definitions[1]: • Work records • Elements • Code and edit tables (to translate and verify data)[2] • Process modules

[1] At some sites, the application developer is not responsible for creating data definitions. [2] For information on code and edit tables, refer to *CA-IDMS Mapping Facility*.

B.2 CA-ADS development tools

ADSA, ADSC, MAPC, and the IDD menu facility all display definition screens to help you define application components. Prompts and default values are displayed whenever possible. When using definition screens, you can:

- **Type specifications** in fields on the screen.

  Specification fields typically precede or immediately follow prompts. For example, when naming an application on the ADSA Main Menu screen:

  ```
  Application name . . . .  xxxappl
                               ↑
                  You type the application name in the
                  specification field that follows the
                  Application name prompt.
  ```

  You enter these specifications when you press [Enter] or any valid PF key (such as [PF5]).

- **Move the cursor from prompt to prompt** by using the forward tab key, backward tab key, or return key. Additionally, you can use any of the cursor movement keys to move the cursor up, down, left, or right.

- **Move from screen to screen** by selecting an activity from the Main Menu or pressing a control key:

  - **You can select an activity** from the Main Menu.

    For example, on the ADSA Main Menu screen, you can select the Task Codes screen by entering **4** opposite the **Screen** prompt.

  - **You can press a control key** that is defined for use in the current development tool.

    For example, **[PF5]** is used to display the next screen in the definition process.

You can invoke an CA-ADS development tool from CA-IDMS/DC or CA-IDMS/UCF (DC/UCF) or from another CA-ADS development tool. You can exit from a development tool at any time. Invoking and exiting from development tools are discussed below.

B.2.1 Invoking development tools

To invoke an CA-ADS development tool, you must be signed on to DC/UCF. The method you use to invoke a development tool depends on your current location:

- **From DC/UCF**, you use the task code defined for the development tool.

  For example, if ADSAT is the task code for the CA-ADS application compiler, you **invoke ADSA from CA-IDMS/DC**:

  ```
  ENTER NEXT TASK CODE:
  adsat
                                                        [Enter]
  ```

Using the task code ADSAT means that you are invoking ADSA under TCF (the transfer control facility). Once you are in the ADSA tool under TCF, you can switch to another application development tool without returning to DC/UCF.

Note: Using the task code **ADSA**, rather than ADSAT, also invokes ADSA. However, TCF is not involved and, because ADSA is not running under TCF, you are not able to switch to another application development tool without returning to DC/UCF first.

►► For more information on task codes, see 6.3, "Application development tools" on page 6-7.

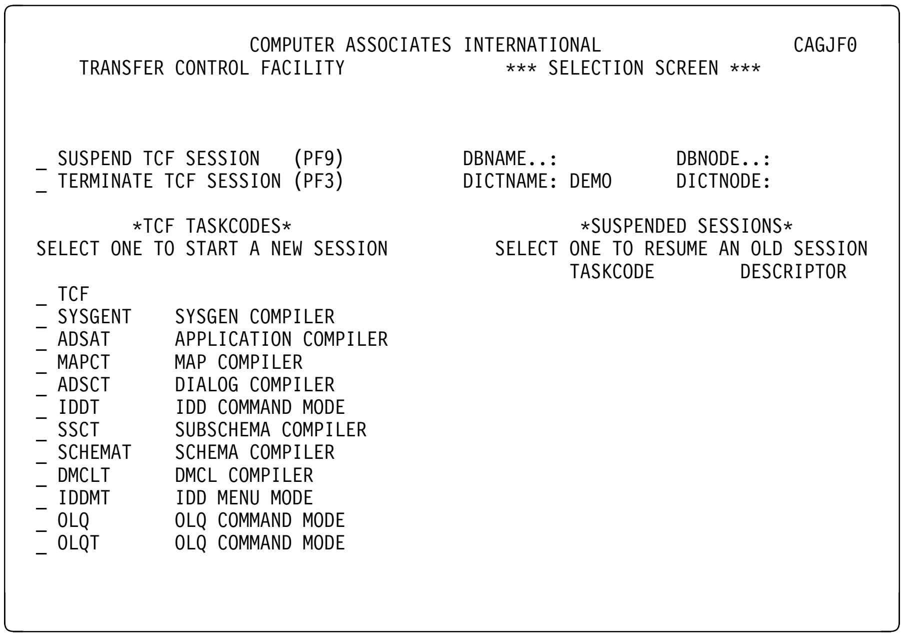
- **From another development tool** under the transfer control facility (TCF), use the command or function key that transfers control from that tool to another. The method used depends on the tool you currently are using:
  - **From ADSA, ADSC, or MAPC**, you can transfer directly to another tool by specifying the task code for the tool after the **Switch** activity on the action bar at the top of the Main Menu.
  - **From IDD**, you can transfer directly to another tool by typing the SWITCH command in the command area at the top of the screen, followed by the task code for that tool.

    For example, if MAPCT is the task code for the online mapping facility (MAPC), you can **transfer from IDD to MAPC**:

    ```
    switch mapct
                                                            [Enter]
    ```

- **From the TCF Selection Screen**, select the development tool's task code from the screen and press [Enter].

  For example, if ADSCT is the task code for the CA-ADS dialog compiler, you can **invoke ADSC from the Selection Screen**:



```
                    COMPUTER ASSOCIATES INTERNATIONAL                  CAGJF0
    TRANSFER CONTROL FACILITY               *** SELECTION SCREEN ***



_ SUSPEND TCF SESSION   (PF9)          DBNAME..:           DBNODE..:
_ TERMINATE TCF SESSION (PF3)          DICTNAME: DEMO      DICTNODE:

         *TCF TASKCODES*                          *SUSPENDED SESSIONS*
SELECT ONE TO START A NEW SESSION         SELECT ONE TO RESUME AN OLD SESSION
                                                 TASKCODE        DESCRIPTOR
_ TCF
_ SYSGENT    SYSGEN COMPILER
_ ADSAT      APPLICATION COMPILER
_ MAPCT      MAP COMPILER
_ ADSCT      DIALOG COMPILER
_ IDDT       IDD COMMAND MODE
_ SSCT       SUBSCHEMA COMPILER
_ SCHEMAT    SCHEMA COMPILER
_ DMCLT      DMCL COMPILER
_ IDDMT      IDD MENU MODE
_ OLQ        OLQ COMMAND MODE
_ OLQT       OLQ COMMAND MODE
```

▸▸ For more information on TCF, refer to *CA-IDMS Transfer Control Facility*.

B.2.2 Exiting from development tools

You can exit from ADSA, ADSC, MAPC, or the IDD menu facility at any time during a definition session. Exit methods available for these development tools include those listed below:

- **The SWITCH command** (under TCF only) allows you to save the definition currently in progress when you exit from a development tool.

  You can use SWITCH to:

  – **Transfer** to another development tool as described above in B.2.1, “Invoking development tools” on page B-4.

  – **Exit** directly to DC/UCF:

    — **From ADSA, ADSC, or MAPC**, specify the SUSPEND keyword after the SWITCH TASK activity at the bottom of the screen:

    ```
    x SWITCH TASK: suspend
                                                              [Enter]
    ```

    — **From IDD**, type the SUSPEND keyword after the SWITCH command in the command area at the top of the screen:

    ```
    switch suspend
                                                              [Enter]
    ```

  ▸▸ For more information on using the SWITCH command, refer to *CA-IDMS Transfer Control Facility*.

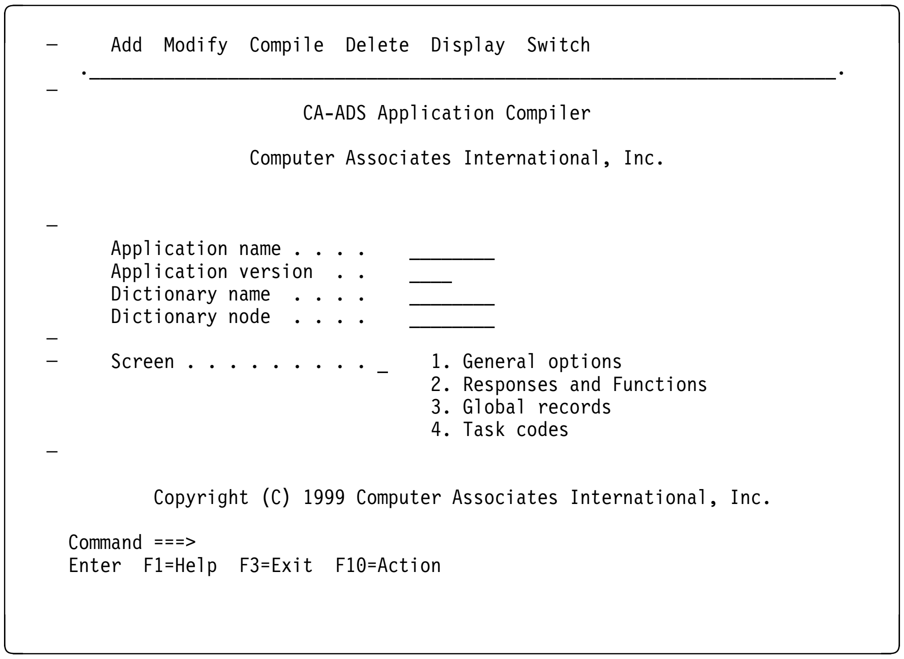
- **The [PF3] key** takes you backward through definition screens until you exit from the tool.

In **IDD**, the SUSPEND and QUIT commands allow you to exit. You type SUSPEND in the screen's command area. Neither SWITCH nor QUIT allows you to specify a task code (for example, IDDMT) or any keywords (for example, the SUSPEND keyword).

B.3 Using ADSA

ADSA is an application design and prototyping tool used to define the structure of an application. The first screen in an ADSA session is the Main Menu screen.

ADSA Main Menu screen



```
 _    Add  Modify  Compile  Delete  Display  Switch
   .______________________________________________________________________.
 _
                      CA-ADS Application Compiler

                 Computer Associates International, Inc.


 _
      Application name . . . .    ________
      Application version  . .    ____
      Dictionary name  . . . .    ________
      Dictionary node  . . . .    ________
 _
 _    Screen . . . . . . . . . _    1. General options
                                    2. Responses and Functions
                                    3. Global records
                                    4. Task codes
 _

          Copyright (C) 1999 Computer Associates International, Inc.

   Command ===>
   Enter  F1=Help  F3=Exit  F10=Action
```

Enter information about the dialog after prompts in the Specification area. **To get from one ADSA screen to another**, you can either select the activity from the Screen Specification area on the Main Menu or, from other screens, press [PF5] to proceed through the definition.

Select an action by tabbing to the action bar or selecting with the command line.

The following table describes how to use ADSA to perform the following procedures:

- Adding an application
- Modifying an application
- Deleting an application
- Adding a response
- Modifying a response
- Deleting a response
- Adding a function

- Modifying a function
- Deleting a function
- Adding a task code
- Modifying a task code
- Deleting a task code

▸▸ For more information on these and other ADSA procedures, refer to *CA-ADS Reference*.

Instructions in the table assume that you have already invoked ADSA, as discussed earlier in this appendix.

Operation	Procedure
Adding an application Procedure to add a new application structure (including responses, functions, and task codes). To define dialogs, maps, and process modules, see descriptions of ADSC, MAPC, and IDD later in this appendix.	1. **Enter basic information** about the application on a Main Menu screen: ■ An application name ■ A dictionary name/node (where applicable) 2. **Select the Add activity from the action bar** to register the application in the dictionary and check it out (reserve it) to the programmer. 3. **Add required components** on appropriate ADSA screens: a. Add **responses and functions** by using the Response/Function List screen as described under "Adding Responses and Functions" later in this table. b. Further define **responses** by using the Response Definition screen, as described in "Adding a Response" later in this table. c. Further define **functions** by using the appropriate Function Definition screen, as described in "Adding a Function" later in this table. d. Add **task codes** by using the Task Codes screen, as described in "Adding a Task Code" later in this table. 4. **Optionally make additional specifications** on other ADSA screens. ▸▸ For more information on available ADSA screens, see the *CA-ADS Reference.* 5. **Create a load module for the application** by selecting the **Compile** activity from the action bar on the Main Menu. For an example of using ADSA to add an application, see Chapter 7, “Defining an Application Structure Using ADSA” on page 7-1.

Operation	Procedure
Modifying an application Procedure to modify an existing application structure (including responses, functions, and task codes). This procedure cannot be used to modify an application name1. To modify dialogs, maps, and process modules, see descriptions of ADSC, MAPC, and IDD later in this appendix.	1. **Display the application definition** (if not already displayed) by entering the following information on the application Main Menu screen: ■ The application's name ■ A dictionary name/node (when applicable) 2. If the application has been released (after having been added), **check the application out** through the **Check out** option under the **Modify** activity on the action bar. 3. **Modify application specifications**, as necessary, on appropriate ADSA screens: a. Add and select responses and functions by using the Response/Function List screen, as described later in this table. b. Add, modify, or delete **responses** by using the Response Definition screen, as described later in this table. c. Add, modify, or delete **functions** by using the Function Definition screens, as described later in this table. d. Add, modify, or delete **task codes** by using the Task Codes screen, as described later in this table. 4. **Recompile the application load module** by selecting the **Compile** activity from the action bar on the Main Menu. For an example of using ADSA to modify an application, see Chapter 10, “Modifying the Application Structure Using ADSA” on page 10-1.
Deleting an application Procedure to delete an application structure (including responses, functions, and task codes). To delete dialogs, maps, and process modules, see descriptions of ADSC, MAPC, and IDD later in this appendix.	1. **Enter the following information** on the application Main Menu screen: a. The application name b. A dictionary name/node (when applicable) 2. **Choose the Delete application option** from the **Delete** activity on the action bar on the Main Menu. 3. **After you press [Enter]**, ADSA displays a confirmation window so that the request to delete the application can be confirmed or rescinded. 4. **Confirm or reject the deletion**. For an example of using ADSA, see Chapter 7, “Defining an Application Structure Using ADSA” on page 7-1.

Operation	Procedure
Adding response and function relationships Procedure to add an application response and function relationships to an application. Do not confuse application responses with dialog response processes2.	1. **Display the Response/Function List screen** by entering a **2** opposite the **Screen** prompt on the Main Menu. 2. **Enter the following information** on the Response/Function List screen: a. The response name b. An associated activity key (optional) c. The name of the function invoked by the response3 d. The type of function (declaration, premap, response, default response) e. The program or dialog name f. A nonblank character to indicate which responses and functions need further definition For an example of using ADSA to add responses and functions, see Steps 4 through 6 in Chapter 7, “Defining an Application Structure Using ADSA” on page 7-1 of this manual.
Enhancing the response definition Procedure to further define an application response to an application. Do not confuse application responses with dialog response processes.	1. **Select responses for further definition** by entering a nonblank character next to the responses on the Response/Function List screen 2. **Display the Response Definition screen** by pressing [PF5] from the Response/Function List screen 3. **Enter the following information** on the Response Definition screen: a. The response type (local or global) b. The control command (optional) used to invoke the associated function For an example of using ADSA to further define responses, see Steps 4 and 5 in Chapter 7, “Defining an Application Structure Using ADSA” on page 7-1 of this manual.

Operation	Procedure
Modifying a response Procedure to modify an application response. This procedure cannot be used to modify a response name4. Do not confuse application responses with dialog response processes5.	1. **Display the responses and functions** on the Response/Function List screen. 2. **Make any changes required**. 3. **Optionally select the response to be modified** by placing a nonblank character next to the response and pressing [PF5] to access the Response Definition screen. 4. **Modify any specifications** on the Response Definition screen, including: a. The response type (global or local) b. The control command used to invoke the function For an example of using ADSA to modify a response, see Steps 2 and 3 in Chapter 10, “Modifying the Application Structure Using ADSA” on page 10-1.
Deleting a response Procedure to delete an application response from an application. Do not confuse application responses with dialog response processes6.	1. **Display the responses and functions** on the Response/Function List screen. 2. **Select the response to be deleted** by placing a nonblank character next to the response and pressing [PF5] to access the Response Definition screen. 3. **Display the response definition** (if not already displayed) on the Response Definition screen as described above in "Modifying a response" earlier in this table. 4. **Place a nonblank character next to the Drop prompt** For an example of using the Response Definition screen, see "Step 5: Further define application responses" in Chapter 7, “Defining an Application Structure Using ADSA” on page 7-1.
Enhancing the dialog function definition Procedure to further define a dialog function to an application7.	1. **Display the responses and functions** on the Response/Function List screen. 2. **Select the function to be defined** by placing a nonblank character next to the function and pressing [PF5]. 3. **Display the Function Definition (Dialog) screen** by pressing [PF5] from the Response/Function List screen. 4. **Enter the following information** on the Function Definition screen: a. A description of the function b. Valid responses for this function. For an example of using ADSA to further define functions see "Step 6: Further define application functions in Chapter 7, “Defining an Application Structure Using ADSA” on page 7-1 of this manual.

Operation	Procedure
Enhancing the menu function definition	1. **Display the responses and functions** on the Response/Function List screen.
Procedure to further define a menu function to an application.[7]	2. **Select the menu function to be defined** by placing a nonblank character next to the function and pressing [PF5].
	3. **Display the Function Definition (Menu) screen** by pressing [PF5] from the Response/Function List screen.
	4. **Enter the following information** on the Function Definition screen:
	a. A description of the function
	b. Specify heading text to be displayed on the runtime menu:
	1) Enter the number of heading lines after the **Heading lines** prompt.
	2) Enter the heading text in lines below the **Heading line text** prompt.
	5. Identify the valid responses for this function.
	a. Display the second page of the **Function Definition (Menu)** screen by pressing [PF8] from page 1.
	b. Enter a nonblank character next to each response that is valid for the menu.
	6. **Customize the menu display** for a menu function[9]:
	a. **Optionally change the way in which responses are displayed** on the runtime menu:
	1) Optionally change the sequence of responses by typing new sequence numbers for the responses to be moved.
	2) Optionally suppress the display of a valid response by entering zeros over the sequence number for the response; the response remains available from the menu even though it is not displayed.
	For an example of changing the response sequence for a menu function, see "Menu functions" in Chapter 7, "Defining an Application Structure Using ADSA" on page 7-1.
	For an example of using ADSA to further define functions see "Step 6: Further define application functions in Chapter 7, "Defining an Application Structure Using ADSA" on page 7-1 of this manual.

Operation	Procedure
Modifying a function Procedure to modify an application function. This procedure cannot be used to modify a function name[10].	1. **Display the responses and functions** on the Response/Function List screen. 2. **Make any changes required**. 3. **Optionally select the function to be further modified** by placing a nonblank character next to the function and pressing [PF5] to access the Function Definition screen. 4. **Modify any specifications** on the Function Definition screen, including: ■ Description ■ Associated dialog ■ Default response ■ Valid responses For an example of using ADSA to modify a function, see "Step 4: Modify the ADDDEP function" in Chapter 10, “Modifying the Application Structure Using ADSA” on page 10-1.
Deleting a function Procedure to delete a function and all responses that invoke that function.	1. **Display the function definition** (if not already displayed) on the Function Definition screen as described above in "Modifying a function" earlier in this table. 2. **Place a nonblank character next to the Drop prompt** For an example of using the Function Definition screen, see "Step 6: Further define application functions" in Chapter 7, “Defining an Application Structure Using ADSA” on page 7-1.
Adding a task code Procedure to add an entry point to the application.	1. **Display the Task Codes screen** by selecting **5** from the Main Menu. 2. **Add the task code:** a. Enter a task code below the **Task code** heading b. Enter a function name below the corresponding **Function** heading. For an example of using ADSA to define a task code, see "Step 7: Define a task code" in Chapter 7, “Defining an Application Structure Using ADSA” on page 7-1.
Modifying a task code Procedure to modify the name or entry-point function for an application.	1. **Display the Task Codes screen** by selecting **5** from the Main Menu. 2. **Erase the task code from the screen** by pressing the ERASE EOF key or typing spaces over the task code. For an example of using the Task Codes screen, see "Step 7: Define a task code" in Chapter 7, “Defining an Application Structure Using ADSA” on page 7-1.

Notes:

1 To modify an application name:

> Add the application definition to the data dictionary, this time using the new name (for instructions, see "Adding an application" earlier in the table) and copy the existing definition using the **Copy** option under the **Add** activity on the action bar of the Main Menu.
>
> Delete the old application, as described in "Deleting an application".

2 To add a response process to a dialog, see "Using ADSC" later in this appendix.

3 Note that ADSA automatically adds a skeleton function with the specified name if the function does not already exist.

4 To modify a response name:

> Add the response definition again, this time using the new name (for instructions, see "Adding a response" earlier in the table).
>
> Delete the old response, as described in "Deleting a response" later in the table.

5 To modify a dialog response process, which is a module of process-language code, see "Using ADSC" later in this appendix.

6 To delete a response process from a dialog, see "Using ADSC" later in this appendix.

7 For **system functions** (such as QUIT), you do not need to follow this procedure; system functions are added to the application automatically when you define responses to invoke the system functions.

8 In addition to the listed function types, you also specify the function type for either a user program or menu/dialog function.

►► For more information on these function types, refer to *CA-ADS Reference*.

9 Valid also for menu/dialog functions.

►► For more information, refer to *CA-ADS Reference*.

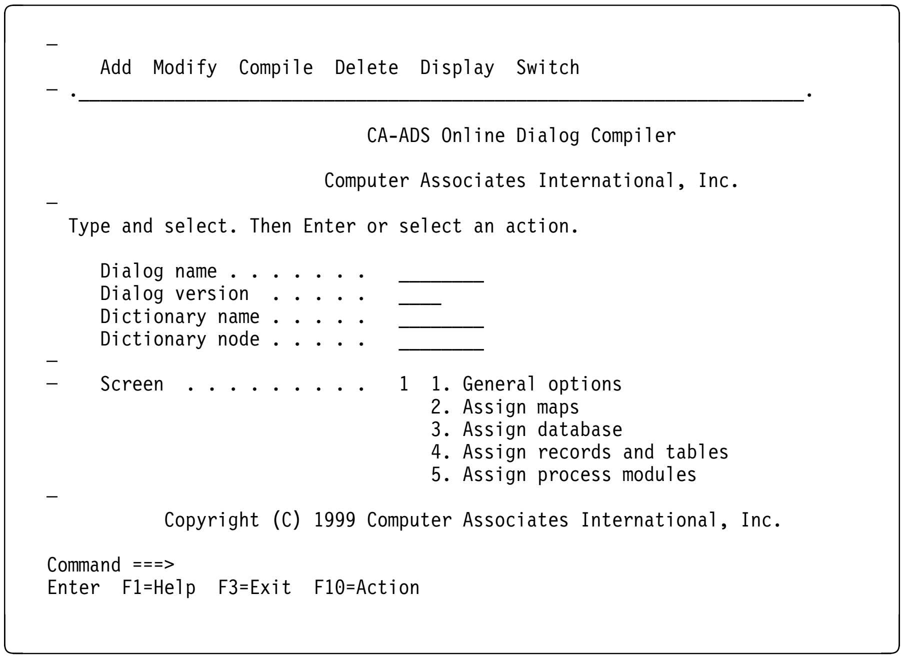
10 To modify a function name:

> Make a note of specifications for the function on the Function Definition screen.
>
> Delete the function as described in "Deleting a function" earlier in the table.
>
> Add the function using the new name (see "Adding a function" earlier in the table.

B.4 Using ADSC

ADSC is the application development tool used to define dialogs. Dialogs are associated with dialog functions in applications. At runtime, a dialog interacts with the user by means of a map (screen). A dialog also performs processing, such as updating the database.

The first screen in an ADSC session is the Main Menu screen. A sample Main Menu screen is shown below:



```
_
    Add  Modify  Compile  Delete  Display  Switch
_ .____________________________________________________________________.
                              CA-ADS Online Dialog Compiler

                          Computer Associates International, Inc.
_
  Type and select. Then Enter or select an action.

     Dialog name . . . . . . .   ________
     Dialog version  . . . . .   ____
     Dictionary name . . . . .   ________
     Dictionary node . . . . .   ________
_
_    Screen  . . . . . . . . .   1  1. General options
                                    2. Assign maps
                                    3. Assign database
                                    4. Assign records and tables
                                    5. Assign process modules
_
           Copyright (C) 1999 Computer Associates International, Inc.

Command ===>
Enter  F1=Help  F3=Exit  F10=Action
```

Enter information about the dialog after prompts in the Specification area.

Select the next ADSC activity or screen from **Screen** prompt.

To get from one ADSC screen to another, you can either enter the number associated with the screen on the Main Menu (and press [Enter]) or press [PF5] to move through the sequence of screens.

Select an action by tabbing to the action bar or selecting with the command line.

What you can do: You can use ADSC to perform the following procedures:

- Adding a dialog
- Modifying a dialog
- Deleting a dialog
- Adding a work record to a dialog

- Modifying a work record in a dialog
- Deleting a work record from a dialog
- Adding a declaration module to a dialog
- Modifying a declaration module specification
- Deleting a declaration module from a dialog
- Adding a premap process to a dialog
- Modifying a premap process specification
- Deleting a premap process from a dialog
- Adding a response process to a dialog
- Modifying a response process specification
- Deleting a response process from a dialog
- Viewing process module errors
- Correcting syntax errors
- Correcting discrepancies between dialog components

The following table describes how to use ADSC to perform the procedures listed above. Instructions in the table assume that you have already invoked ADSC, as discussed in B.2.1, “Invoking development tools” on page B-4 earlier in this appendix.

►► For more information on these and other ADSC procedures, refer to *CA-ADS Reference*.

Operation	Procedure
Adding a dialog Procedure to define a skeleton dialog (for a prototype application) or a fully functional dialog.	1. **Enter basic information** about the dialog on the dialog Main Menu screen: a. The dialog name b. A dictionary name/node (when applicable) 2. **Associate a map with the dialog** on the Map Specifications screen. 3. **Associate a subschema or access module with the dialog** on the Database Specifications screen.1 If you are creating a skeleton dialog, you can now proceed immediately to Step 5 to compile the dialog. 4. **Optionally make additional specifications** on appropriate ADSC screens: a. Specify additional **dialog options** (for example, symbol and diagnostic tables to simplify the application development process) by using the Options and Directives screen. For more information on the Options and Directives screen, see "Step 2: Specify dialog options" in Chapter 12. b. Associate one or more **work records** or SQL-defined tables with the dialog by using the Records and Tables screen, as described in "Adding a work record to a dialog" later in this table. c. Associate a **declaration module**, **premap process**, and one or more **response processes** with the dialog by using the Process Modules screen, as described in "Adding processes to a dialog" later in this table. 5. **Create a dialog load module** by selecting the **Compile** activity from the action bar on the Main Menu. For an example of using ADSC to add dialogs, see Chapter 9, “Defining Dialogs Using ADSC” on page 9-1.

Operation	Procedure
Modifying a dialog Procedure to modify a dialog definition. This procedure cannot be used to modify a dialog name2.	1. **Display the dialog definition** (if not already displayed) by entering the following on the Main Menu screen: a. The dialog name b. the dictionary name/node (when applicable) 2. **Modify dialog specifications** on appropriate ADSC screens: a. Optionally add or modify **dialog options** by using the Options and Directives screen. For more information on the Options and Directives screen, see "Step 2: Specify dialog options" in Chapter 12. b. Optionally add or modify **map information** on the Map Specifications screen. For more information on the Map Specifications screen, see "Step 3: Name the associated map" in Chapter 9. c. Optionally add or modify **subschema or access module information** on the Database Specifications screen. For more information on the Database Specifications screen, see "Step 3: Add a subschema" in Chapter 12. d. Optionally add, modify, or delete **work record specifications** or **tables** on the Records and Tables screen, as described later in this table. e. Optionally add, modify, or delete the **process module specifications** on the Process Modules screen, as described later in this table. 3. **Recompile the dialog load module** by selecting the **Compile** activity on the Main Menu. For examples of modifying dialogs, see "Updating modified maps in dialogs using ADSC" in Chapter 11 and "Updating modified process modules in dialogs using ADSC" in Chapter 13.
Deleting a dialog Procedure to delete a dialog from the data dictionary.	1. **Enter the following information** on a blank Main Menu screen: a. The dialog name b. the dictionary name/node (when applicable) 2. **Choose the Delete activity** from the action bar. 3. **Confirm or rescind** the deletion. For an example of using ADSC, see Chapter 9, “Defining Dialogs Using ADSC” on page 9-1.

Operation	Procedure
Adding a work record to a dialog Procedure to make an existing work record or table available to a dialog and its processes[3]. To add a work record to the data dictionary, see "Using IDD" later in this appendix.	1. **Display the Records and Tables screen** by selecting **4** from the Main Menu screen. 2. **Enter the name of the work record** or **table** to be associated with the dialog under the **Record name** prompt. 3. If this is a work record, enter a nonblank character under the prompt **Work**. For an example of adding a work record to a dialog, see "Step 3: Add a work record" in Chapter 15.
Deleting a work record from a dialog Procedure to delete a work record or table from a dialog. This procedure does not delete a work record from the data dictionary[4].	1. **Display the Records and Tables screen** by selecting **4** from the Main Menu screen. 2. **Enter a nonblank character in the Drop column** opposite the work record or table to be deleted. For an example of how to use the Records and Tables screen, see "Step 3: Add a work record" in Chapter 15.

Operation	Procedure
Adding process modules to a dialog Procedure to associate existing process modules with a dialog. To define a process module in the data dictionary, see "Using IDD" later in this appendix.	1. **Display the Process Modules screen** by selecting **5** from the Main Menu. 2. **Enter the name of the process module** to be associated with the dialog after the **Name** prompt. 3. **Specify the type** of process module it is: declaration, premap, response, default response. 4. **If this is a response process**, enter the following information: a. **A control key** (optional) specified after the **Key** prompt. b. **A response field value** (optional) specified after the **Value** prompt **Note:** Each response process *must* have a key and/or a response field value, or be the default response. c. **Execution status on input errors** (optional) enabled by entering a nonblank character before the **Execute on edit errors** prompt; in this case, the response process is executed at runtime even when input errors are entered. ►► For information on automatic editing, refer to *CA-IDMS Mapping Facility*. 5. **Press [Enter]** to add the process module to the dialog definition. For an example of using ADSC to associate a process module with a dialog, see "Adding process modules to dialogs using ADSC" in Chapter 12. ►► For more information on the Process Module screen, refer to *CA-ADS Reference*.
Modifying a process module Procedure to associate a modified process module with a dialog or to replace the current process module. This procedure does not modify source statements for the process module in the data dictionary5.	1. **Display the Process Modules screen**. 2. **Optionally change the process module name** by typing over the displayed name. The new name must identify an existing process module in the data dictionary. 3. **Optionally modify specifications**: ■ Key ■ Value ■ Execute on errors ■ Type (1=Declaration, 2=Premap, 3=Response, 4=Default response) 4. **Press [Enter]** to input modified Process Modules screen specifications. For an example of modifying a process module specification, see "Updating modified process modules in dialogs using ADSC" in Chapter 13.

Operation	Procedure
Deleting a process module from a dialog Procedure to delete a process module from a dialog. This procedure does not delete the process module from the data dictionary.6	1. **Display the Process Modules screen** (if it not already displayed) by selecting **5** from the Main Menu. 2. **Enter a nonblank character next to the Drop prompt**. For an example of using the Process Modules screen, see "Adding process modules to dialogs using ADSC" in Chapter 12.
Compiling the dialog Procedure to compile process code and create the dialog load module.	1. **Create a load module for the application** by selecting the **Compile** activity from the action bar on the Main Menu. ADSC will attempt to compile all process modules associated with the dialog and, if successful, create a load module.10 **Note: If ADSC finds errors** during the compile, the process module is not added to the dialog. ADSC displays an error message. In this case, you must view and correct process module errors and then recompile the process module. For more information, see "Viewing process module errors" later in this table.
Viewing process module errors Procedure to view compile-time errors and error messages found by ADSC in a process module. To correct errors in process modules, see "Correcting syntax errors" and "Correct discrepancies between dialog components" later in this table.	1. **Display a listing of the module and its errors:** a. Choose the **View messages** option from the **Compile** activity on the action bar on the Main Menu.
Correcting syntax errors Procedure to correct syntax errors (such as omitted periods or misspelled words) in a process module. To view error messages for a process module, see "Viewing process module errors" earlier in this table.	1. Choosing the **View messages** option from the **Compile** activity on the action bar of the Main Menu will bring you to the Compiled Process Modules screen. 2. On the Compiled Process Modules screen, choose **Display** to go to the Dialog Process Source screen. The Dialog Process Source screen will show you the process source and errors encountered by the compiler. 3. Press [PF5] to move to IDD to correct errors.11 For an example of using IDD and ADSC to correct syntax errors, see "Correct syntax errors" in Chapter 12.

Operation	Procedure
Correcting discrepancies between dialog components Procedure to correct discrepancies between a process module and another dialog component (for example, a map, record, record element, or subschema). To view error messages for a process module, see "Viewing process module errors" earlier in this table.	1. Choosing the **View messages** option from the **Compile** activity on the action bar of the Main Menu will bring you to the **Structural Error Display** screen. 2. **Correct the errors** by using the appropriate development tool: ■ **For errors made on ADSC screens** (such as an incorrect subschema name), proceed to the appropriate screen and change the specification. ■ **For errors made in a different development tool** (such as a record definition made by using the IDD menu facility): a. Transfer to the development tool by selecting the **Switch** activity from the action bar on the Main Menu (if you are operating under TCF) and specify the task code for the development tool. For more information on transferring to other development tools, see "Invoking development tools" earlier in this appendix. b. Correct and recompile the definition: 1) To modify **maps**, see "Using MAPC" later in this appendix. 2) To modify **work records, elements,** and **process modules**, see "Using IDD" later in this appendix. c. **Transfer back to ADSC** (if necessary) by using the **Switch** activity. 3. **Recompile the dialog**. For a discussion of correcting discrepancies between dialog components, see "Correct discrepancies" in Chapter 12.

Notes:

1 You do not have to name the subschema and schema in the following situations:

- If you are creating a skeleton dialog for the basic prototype of an application.
- If the dialog uses only SQL statements to access the database
- If the dialog does not access a database

An access module needs to be named only if the dialog is accessing a database using SQL statements *and* an existing access module is going to be used.

2 To modify a dialog name, copy the dialog definition to a new dialog name using the **Copy** option of the **Add** activity on the action bar on the Main Menu. Delete the old dialog.

3 Work records associated with the dialog's map are *automatically* available to the dialog and do not need to be added to the dialog separately.

4 To delete a work record from the data dictionary, see "Using IDD" later in this appendix.

5 To modify process module source statements in the data dictionary, see "Using IDD" later in this appendix.

6 To delete a process module from the data dictionary, see "Using IDD" later in this appendix.

7 To define a process module in the data dictionary, see "Using IDD" later in this appendix.

8 You can associate a response process with several control keys and/or response field values. To do this, add the response process to the dialog several times, each time specifying a different control key and/or response field value for the response process. Only one copy of the compiled response is included in the dialog load module.

9 To modify process module source statements, see "Using IDD" later in this appendix.

10 Whenever you select the **Compile** activity, ADSC compiles all process modules before creating a load module for the dialog.

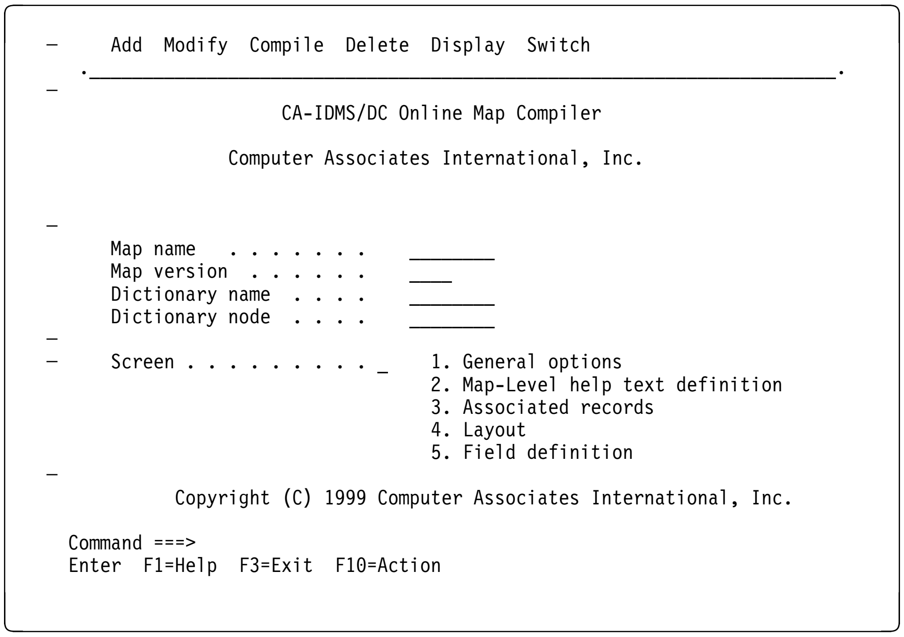
11 When you switch to IDD from ADSC after encountering compile errors, you are in full-screen mode in IDD.

►► For more information on IDD full-screen mode, refer to *CA-IDMS Online Compiler Text Editor*.

B.5 Using MAPC

MAPC is the application development tool used to define online maps. In CA-ADS applications, maps are displayed by dialogs.

The first screen in an MAPC session is the Main Menu screen. A sample Main Menu screen is shown below:



```
 _     Add  Modify  Compile  Delete  Display  Switch
    .______________________________________________________________________.
 _
                      CA-IDMS/DC Online Map Compiler

                 Computer Associates International, Inc.


 _
       Map name   . . . . . . .    ________
       Map version  . . . . . .    ____
       Dictionary name  . . . .    ________
       Dictionary node  . . . .    ________
 _
 _     Screen . . . . . . . . . _    1. General options
                                     2. Map-Level help text definition
                                     3. Associated records
                                     4. Layout
                                     5. Field definition
 _
            Copyright (C) 1999 Computer Associates International, Inc.

   Command ===>
   Enter  F1=Help  F3=Exit  F10=Action
```

You can display an MAPC screen by entering a value opposite the **Screen** prompt and pressing [Enter].

The following table describes how to use MAPC to perform the following procedures:

- Adding a map
- Modifying a map
- Deleting a map
- Adding new fields to a map
- Modifying existing fields on a map
- Deleting fields from a map

▸▸ For more information on these and other MAPC procedures, refer to *CA-IDMS Mapping Facility*.

Instructions in the table assume that you have already invoked MAPC, as discussed earlier in this appendix.

Operation	Procedure
Adding a map Procedure to add a map definition to the data dictionary To add an existing map to a dialog, see "Using ADSC" earlier in this appendix.	1. **Specific basic information** about the map on the MAPC Main Menu: a. The map name b. The dictionary name/node (when applicable) 2. **Names the associated database and work records** on the Associated Records screen. 3. **Autopaint the map** by naming the map fields on the Automatic Screen Painter screen. 4. **Optionally make additional specifications** on other MAPC screens. ▶▶ For more information on available MAPC screens and definition options, refer to *CA-IDMS Mapping Facility*. 5. **Optionally modify the map layout** using the Layout screen. 6. **Create a map load module** using the **Compile4** activity from the action bar on the Main Menu. For an example of using MAPC to add a map definition, see Chapter 8, “Defining a Screen Display Using MAPC” on page 8-1.

Operation	Procedure

Modifying a map
 	
Procedure to modify an existing map.
 	
This procedure cannot be used to modify a map name[1].
 	
To update a modified map definition in a dialog, recompile the dialog as described in "Using ADSC" earlier in this appendix.
	1. **Display the map definition** by entering on a Main Menu screen:
	a. The map name
	b. the dictionary name/node (when applicable)
	2. **Optionally add and modify basic information** about the map, including names of associated database and work records.
	3. **Optionally add, modify, and/or delete fields.** You can add, modify, and delete fields at the same time:
	■ To add new fields, see "Adding new fields to a map" later in this table.
	■ To modify fields, see "Modifying existing fields on a map" later in this table.
	■ To delete fields, see "Deleting fields from a map" later in this table.
	4. **Optionally add, modify, and/or delete specifications** on other MAPC screens.
	▸▸ For more information on available MAPC specifications, see *CA-IDMS Mapping Facility*.
	5. **Recompile the map load module:** by specifying the **Compile**[4] activity from the action bar on the Main Menu.
	Note: If new copies of maps are not automatically loaded in the program pool at your site, return to CA-IDMS/DC or CA-IDMS/UCF and issue the following command before executing the modified map:
	`DCMT VARY PROGRAM modified-map-name`
	`NEW COPY.`
	This command causes a new copy of the map to be loaded in the program pool.
	
	For an example of using MAPC to modify a map, see Chapter 11, "Modifying a Map Using MAPC" on page 11-1.

Operation	Procedure
Deleting a map Procedure to delete a map from the data dictionary. To delete a map from a dialog, modify the dialog as described in "Using ADSC" earlier in this appendix.	1. **Identify the map** to be deleted on the Main Menu screen by entering: a. The map name b. the dictionary name/node (when applicable) 2. **Choose the Delete action[4] from the action bar** 3. **Confirm or rescind the action** The next time you modify a dialog associated with a deleted map, ADSC displays a message warning you that the map has been deleted. Users can continue to execute a deleted map until a new copy of the map is loaded in the program pool, provided that automatic load is not specified in the system generation OLM statement. For an example of using MAPC, see Chapter 8, “Defining a Screen Display Using MAPC” on page 8-1.
Adding new fields to a map Procedure to add new fields to a map. To move or modify fields, see "Modifying existing fields on a map" later in this table. To delete fields, see "Deleting fields from a map" later in this table.	1. **Display the Layout screen** from the Main Menu by entering **2** opposite the **Screen** prompt and pressing [Enter]. 2. **Define each new field** on the Layout screen as described below: a. **Move the cursor** to the screen position that immediately precedes the starting position of the field. b. **Type a start-field character[2]** to signal the start of the field. The field starts in the column immediately after the start-field character. c. **For a literal field only,** specify the value to be displayed at runtime by typing a literal string after the start-field character, as shown below for a field that displays the value DEPARTMENT INFORMATION: `;DEPARTMENT INFORMATION` **Note: For a variable field,** you will specify the value to be displayed at runtime in Step 3 later in this procedure. d. **Optionally add additional new fields** as described in Steps 1, 2, and 3 above. e. **Press [Enter]** to save the current map layout for further definition. 3. **Select fields for further editing** by positioning the cursor on the field and pressing [PF2] or by overtyping the start-field character with the field-select character (the default is a percent sign).[3]

Operation	Procedure
Editing fields Procedure to edit variable fields on a map.	1. **Press [PF5] from the Layout screen** to access the Literal or Field Definition screen. 2. **Edit variable fields** on the Field Definition screen as described below: a. **Specify the value to be displayed by the field** by naming a record element or system-supplied field ($RESPONSE, $MESSAGE, or $PAGE) after the **Element** prompt. b. **Optionally override default specifications** for the variable field: For an example of adding fields to a map, see Chapter 8, “Defining a Screen Display Using MAPC” on page 8-1.
Modifying existing fields on a map Procedure to modify or move existing fields on a map. Adding new fields to a map is presented in "Adding new fields to a map" earlier in this table. To associate the modified map with any dialogs that use the map, you recompile the dialog load module. See "Using ADSC" earlier in this appendix.	1. **Display the Layout screen** from the Main Menu screen by entering **4** at the **Screen** prompt. 2. **Use the alternate set of function keys to mark and move the fields** 3. **Select one or more fields to be modified** by positioning the cursor on the field and pressing [PF2] or by overtyping the start-field character with a percent sign. 4. **Modify each field definition selected** as described below: a. Display the Field Definition or Literal Definition screen. b. Modify any specifications for the field. 5. **Recompile the map load module** if there are no more updates to be made to the map by selecting the **Compile4** activity from the action bar on the Main Menu. For an example of modifying fields on a map, see Chapter 11, “Modifying a Map Using MAPC” on page 11-1.
Deleting fields from a map Procedure to delete existing fields from a map To associate the modified map with any dialogs that use the map, recompile the dialog load module. See "Using ADSC" earlier in this appendix.	1. **Display the Layout screen** by entering **4** at the **Screen** prompt on the Main Menu. 2. **Make sure that insert-character mode is disabled** by pressing the RESET key. 3. **Select the fields to be deleted** by positioning the cursor on the field or fields and pressing [PF2] or by overtyping the start-field character with the field-select character. 4. Press [PF5] to move to the Literal Definition or Field Definition screen. 5. **Enter a nonblank character** next to the **Drop field** prompt to delete the field. 6. **Recompile the map load module** (if there are no more updates to be made to the map) by selecting the **Compile** activity from the action bar on the Main Menu.

Notes:

1 To modify a map name:

Make a copy of the original map, using the new name for the copy.

►► For more information on copying a map, refer to *CA-IDMS Mapping Facility*.

Create a load module for the new copy.

2 The start-field character is defined at system generation time; for example, default start-field characters are the field mark (;) or the left brace (&lbr.).

3 The select-field and start-field characters are defined at system-generation time and can vary from site to site; the default select-field character is the percent sign (%).

4 The dictionary is updated only on a **Compile** or a **Delete** action. A record becomes map-owned only after the map is compiled.

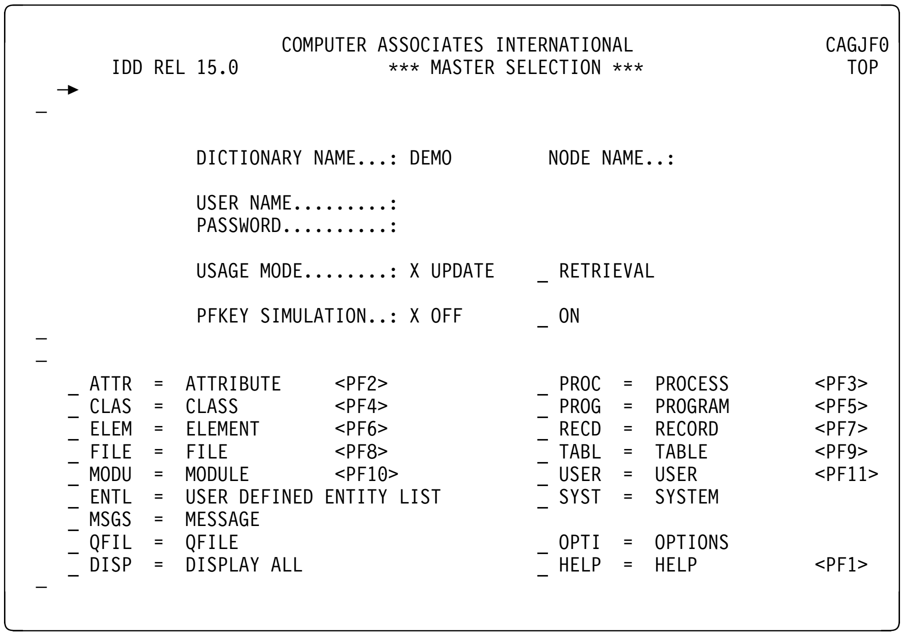
B.6 Using the IDD Menu Facility

As an application developer, you can use the **IDD menu facility** to define data (records and elements) and process modules in the data dictionary.

Alternatively, you can use **online IDD** to define data and process modules.

▸▸ For information on how to use online IDD, see the *IDD DDDL Reference*.

The first screen in an IDD menu facility session is the Master Selection screen. A sample Master Selection screen is shown below:



```
                          COMPUTER ASSOCIATES INTERNATIONAL                  CAGJF0
          IDD REL 15.0              *** MASTER SELECTION ***                    TOP
    →
  _


                  DICTIONARY NAME...: DEMO          NODE NAME..:

                  USER NAME.........:
                  PASSWORD..........:

                  USAGE MODE........: X UPDATE     _ RETRIEVAL

                  PFKEY SIMULATION..: X OFF        _ ON
  _
  _
     _ ATTR  =  ATTRIBUTE     <PF2>               _ PROC  =  PROCESS        <PF3>
     _ CLAS  =  CLASS         <PF4>               _ PROG  =  PROGRAM        <PF5>
     _ ELEM  =  ELEMENT       <PF6>               _ RECD  =  RECORD         <PF7>
     _ FILE  =  FILE          <PF8>               _ TABL  =  TABLE          <PF9>
     _ MODU  =  MODULE        <PF10>              _ USER  =  USER           <PF11>
     _ ENTL  =  USER DEFINED ENTITY LIST          _ SYST  =  SYSTEM
     _ MSGS  =  MESSAGE
     _ QFIL  =  QFILE                             _ OPTI  =  OPTIONS
     _ DISP  =  DISPLAY ALL                       _ HELP  =  HELP           <PF1>
  _
```

To get from one IDD screen to another, you can enter the identifier for the next screen in the current screen's command area, select the screen identifier from the Activity Selection area, or press the control key associated with the screen. Identifiers and control keys for available screens are listed in the Activity Selection area of IDD menu facility screens.

The following table describes how to use the IDD menu facility to perform the following procedures:

- Adding an element
- Modifying an element
- Deleting an element
- Adding a work record

- Modifying a work record
- Deleting a work record
- Adding a process module
- Modifying a process module
- Copying a process module
- Deleting a process module

▶▶ For more information on using the IDD menu facility, refer to *CA-IDMS Online Compiler Text Editor*.

▶▶ For information on other operations you can perform by using IDD, refer to *IDD DDDL Reference*.

Instructions in table assume that you have already invoked IDD, as discussed earlier in this appendix.

Operation	Procedure
Adding an element Procedure to add a new element to the data dictionary To add an existing element to a work record, see "Adding a work record" later in this table.	1. **Display the Element Entity screen** by entering the identifier (**elem**) for the screen in the command area, as shown below[1]: `--▶ elem` 2. **Define the element** on the Element Entity screen: a. Type an element name. b. Select the ADD action and deselect (space over) DISPLAY. c. Optionally type a description of the element. d. Specify a picture; for example, PIC X(20). e. Select a usage mode; for example, COMP-3 (default is DISPLAY). For an example of using the IDD menu facility to define an element, see "Step 1: Define an element" in Chapter 14.

Operation	Procedure
Modifying an element Procedure to modify an element definition in the data dictionary. To update the modified element in any records that contain the element, see "Modifying a work record" later in this table. This procedure cannot be used to modify an element name2.	1. **Display the element to be modified** on the Element Entity screen by typing the identifier (ELEM) for the screen in the command area, followed by the name of the element, as shown below for an element named ELEMENT1: `→ elem element1` 2. **Modify the element**: a. Select the MODIFY action and deselect DISPLAY. b. Modify any of the following specifications: ■ Description ■ Picture ■ Usage mode For an example of using the Element Entity screen, see "Step 1: Define an element" in Chapter 14.
Deleting an element Procedure to delete an element definition from the data dictionary. This procedure cannot be used to delete an element that already belongs to a work record3.	1. **Display the element to be deleted** on the Element Entity screen by typing the identifier (ELEM) for the screen in the command area, followed by the name of the element, as shown below for an element named ELEMENT1: `→ elem element1` 2. **Delete the element:** a. Select the DELETE action. b. Deselect the DISPLAY action. For an example of using the Element Entity screen, see "Step 1: Define an element" in Chapter 14.

Operation	Procedure

Adding a work record
 	
Procedure to add a work record to the data dictionary.
 	
To add elements to the data dictionary, see "Adding an element" earlier in this table.
	1. **Display the Record Entity screen** by entering the identifier (**recd**) for the screen in the command area:
	`→ recd`
	2. **Define the work record** on the Record Entity screen:
	a. Type a record name.
	b. Select the ADD action and deselect DISPLAY.
	c. Optionally make additional specifications for the work record.
	▶▶ For more information on available record options and screens, refer to *IDD DDDL Reference*.
	3. **Associate existing elements with the work record** by using the Record Element screen:
	a. Display the Record Element screen from the Record Entity screen by entering the identifier (**relm**) for the screen in the command area:
	`→ relm`
	b. Associate an element with the work record:
	1) Type the name of an existing element.
	2) Optionally override any existing element specifications for:
	■ Picture
	■ Usage mode
	3) Optionally make other specifications for the element.
	▶▶ For more information on available record options and screens, refer to *IDD DDDL Reference*.
	c. Optionally associate another element with the work record as described in Step 3.2 above, after first pressing the page-forward key (default is [PF8]) to display a blank Record Element screen.
	For an example of using the Element Entity screen, see "Step 2: Define a work record" in Chapter 14.

Modifying a work record -- continued on next two pages

Procedure to modify a work record in order to replace elements in the record or to modify specifications for record elements.

This procedure cannot be used to modify a work record name4.

1. **Display the work record to be modified** by typing the identifier (RECD) for the Record Entity screen in the command area, followed by the name of the record, as shown below for a record named WK-RECORD1:

   ```
   → recd wk-record1
   ```

2. **Optionally add, modify, or delete work record specifications.**

   ►► For more information on record specifications and screens, refer to *IDD DDDL Reference*.

3. **Optionally modify element specifications** for the record:

   - **To associate an element with the work record:**

     a. Display the Record Element screen by entering the identifier (**relm**) for the screen in the command area:

     ```
     → relm
     ```

     b. Associate an element with the record:

     1) type the name of an existing element5.

     2) Optionally override any existing element specifications for:

        – Picture

        – Usage mode

     3) Optionally make other specifications for the element.

        ►► For more information on record element specifications, refer to *IDD DDDL Reference*.

     c. Optionally associate another element with the record as described in Step b above, after first pressing the page-forward key (default is [PF8]) to display a blank Record Element screen.

   - **To replace an element in the record:**

     a. **Display** the record element specification in either of the following ways:

     – When the record contains a *few* record elements:

       1) Display the Record Element screen by entering the identifier (**relm**) for the screen in the command area:

       ```
       → relm
       ```

       2) Page through record elements in the record (if necessary) to display the required record element. To do this, press the page-forward key (default is [PF8]).

Modifying a work record -- continued

- When the record contains *several* record elements:
  1) Display the Record Element List screen by entering the identifier (**rell**) for the screen in the command area:

     ```
     → rell
     ```

  2) From the displayed list of elements, select the record element to be replaced by entering a nonblank character in the SELECT column for the element.

b. **Modify** the record element specification:

1) Select the REPLACE action.

   Note: The REPLACE action:

   a) Deletes the element from the work record

   b) Adds a new copy of the element to the record.

   REPLACE does not modify the original element in the data dictionary.

2) Optionally type a new element name to replace the existing element with another.

3) Optionally override any existing element specifications for:
   - Picture
   - Usage mode

4) Optionally modify other specifications

   ►► For more information on available record element specifications, refer to *IDD DDDL Reference*.

c. Optionally **select and modify another** record element specification:

1) Return to the Record Element List screen by pressing CLEAR (after pressing [Enter] to make sure that all record element specifications are entered).

2) Select the record element to be modified by entering a nonblank character in the SELECT column for the element.

3) Modify the record element as described in Step b above.

Modifying a work record -- continued

- **To remove an element from the record:**

  a. **Display** the record element specification:

     1) Display the Record Element List screen by entering the identifier (**rell**) for the screen in the command area:

        ```
        → rell
        ```

     2) From the displayed list of elements, select the record element specification to be removed by entering a non-blank character in the SELECT column for the element.

  b. **Remove** the element by selecting the REMOVE action.

     Note: The REMOVE action deletes the record element specification from the record but not from the data dictionary.

For an example of using the IDD menu facility for records, see Chapter 14, "Defining Work Records Using IDD" on page 14-1.

Deleting a work record

Procedure to delete a record from the data dictionary.

This procedure cannot be used to delete a work record used by a map or a dialog[6].

This procedure does not delete record elements from the data dictionary[7].

1. **Display the work record to be deleted** on the Record Entity screen by typing the identifier (RECD) for the screen in the command area, followed by the name of the record to be deleted, as shown below for a record named WK-RECORD1:

   ```
   → recd wk-record1
   ```

2. **Delete the work record**:

   a. Select the DELETE action.

   b. Deselect the DISPLAY action.

For an example of using the IDD menu facility for work records, see Chapter 14, "Defining Work Records Using IDD" on page 14-1.

Adding a process module

Procedure to add a process module definition and source statements to the data dictionary.

To make an existing process module the premap or response process for a dialog, see "Using ADSG" earlier in this appendix.

1. **Display the Process Entity screen** by entering the identifier (**proc**) for the screen in the command area:

   ```
   → proc
   ```

2. **Specify basic information about the process module** on the Process Entity screen:

   a. Type the process module name.

   b. Select the ADD action and deselect DISPLAY.

   c. Optionally type a description for the process module.

3. **Display the Process Source screen** from the Process Entity screen by entering **srce** in the command area:

   ```
   → srce
   ```

4. **Enter process source statements** for the modules.

   a. Type one or more lines of process statements on a page of the Process Source screen.

      Note: Do not extend process source statements beyond column 72.

   b. Optionally enter additional pages of process source statements. For each new page:

      1) Place the cursor on the line after which new source statements are to be inserted.

      2) Press [PF4] (default) to open new lines after the cursor.

      3) Enter lines of process statements on the new page.

      4) Press [PF5] (default) to apply your changes to the work file maintained by the IDD menu facility.

5. **Press [Enter]** to add the process source statements to the process module in the data dictionary.

For an example of using the IDD menu facility to define a process module, see "Defining process modules using IDD" in Chapter 12.

A process module's source statements are compiled when the process module is added to a dialog. For information on adding process modules (as premap and response processes) to a dialog, see "Using ADSC" earlier in this appendix.

Modifying a process module

Procedure to modify a process module in the data dictionary.

This procedure does not update dialogs that use the process module9.

This procedure cannot be used to modify a process module name10.

1. **Display the process module to be modified** on the Process Entity screen by typing the identifier (PROC) for the screen in the command area, followed by the name of the process module, as shown below for a process module named PROCESS1:

   ```
   → proc process1
   ```

2. **Modify process statements**, as necessary:

   a. Display process statements for the module on the Process Source screen by entering the identifier (**srce**) for the screen in the command area:

      ```
      → srce
      ```

   b. Add, modify, and delete process statements. To insert one or more lines of statements:

      1) Place the cursor on the line after which new source statements are to be inserted.

      2) Press [PF4] (default) to open new lines after the cursor.

      3) Enter lines of process statements on the new page.

      4) Press [PF5] (default) to apply the new statements to the work file maintained by the IDD menu facility.

      Note: Do not extend statements beyond column 72.

   c. Optionally page through the Process Source screen:

      - Press [PF8] (default) to page forward.
      - Press [PF7] (default) to page backward.

3. **Press [Enter]** to modify the process module stored in the data dictionary.

For an example of modifying a process module, see Chapter 13, “Modifying Process Logic in a Dialog” on page 13-1.

Copying a process module

Procedure to copy source statements from one process module to a new process module[1].

1. **Display the Process Entity screen** by entering the identifier (**proc**) for the Process Entity screen in the command area:

   ```
   → proc
   ```

2. **Specify basic information about the new process module** on the Process Entity screen:

   a. Type the name of the new process module.

   b. Select the ADD action and deselect DISPLAY.

   c. Optionally type a description for the new process.

3. **Display the Copy screen** by entering the identifier (**copy**) for the screen in the command area:

   ```
   → copy
   ```

4. **Enter process module specifications**:

   a. Type the name of the *original* process module after the COPY FROM PROCESS NAME prompt.

   b. Select the PROCESS TEXT action to copy the process commands in the original process module to the current process module.

5. **Modify process statements** if necessary, by using the Process Source screen as described in "Modifying a process module" earlier in this table.

For an example of using IDD for process modules, see Chapter 12, “Adding Process Logic to a Dialog” on page 12-1.

Deleting a process module

Procedure to delete a process module from the data dictionary.

To delete a process module (premap or response process) from a dialog, see "Using ADSC" earlier in this appendix.

1. **Display the process module to be deleted** by typing the identifier (PROC) for the Process Entity screen in the command area, followed by the name of the process module, as shown below for a process module named PROCESS1:

   ```
   → proc process1
   ```

2. **Delete the process module**:

   a. Select the DELETE action.

   b. Deselect the DISPLAY action.

For an example of using IDD for process modules, see Chapter 12, “Adding Process Logic to a Dialog” on page 12-1

Notes:

[1] You can also display any IDD menu facility screen by using either of the following methods:

- Select the activity that identifies the screen (in this case, ELEM) from the Activity Selection area of the current screen and press [Enter].
- Press the control key listed for the screen (in this case, [PF6]).

2 To modify an element name:

Add the element definition to the data dictionary again, this time using the new name. For instruction, see "Adding an element", earlier in the table.

Add the new element to and remove the old element from any work records that contain the element, as described in "Modifying a work record" later in the table.

Then delete the old element from the data dictionary, as described below in "Deleting an element".

3 To delete an element that belongs to a work record:

Remove the element from the record, as described in "Modifying a work record" later in the table.

Delete the element from the data dictionary.

4 To modify a work record name:

Add the work record definition to the data dictionary again, this time using the new name. For instruction, see "Adding an element", earlier in the table.

Delete the old record from the data dictionary as described in "Deleting a work record" later in the table.

5 If the element already belongs to the record, specifications on the Record Element screen *modify* previous specifications for the record element.

6 If the work record participates in maps and/or dialogs:

Delete the record from each map (by using the MAPC Associated Records screen) and/or from each dialog (by using the ADSC Records and Tables screen).

Delete the record from the data dictionary.

7 To delete elements from the data dictionary, see "Deleting an element" earlier in the table.

8 For information on specific process statements, refer to *CA-ADS Reference*.

9 To update dialogs that use the module, see "Modifying a premap process specification" or "Modifying a response process specification" as appropriate, in "Using ADSC" earlier in this appendix. At this point, the process module's source statements are compiled.

10 To modify a process module name:

Add a new process module, using the new name, on the Process Entity screen.

Copy the source statements from the original process module by using the Copy screen, as described in "Copying a process module" later in the table.

If appropriate, delete the original process module from the data dictionary, as described in "Deleting a process module" later in the table.

11 You can also *use* one process module's source statements in another process module by using the INCLUDE command. The INCLUDE command names the process module whose source statements are included in the current process module at compile time.

Appendix C. Layout of the DEPARTMENT Record

This appendix presents the layout and configuration of the sample DEPARTMENT record. DEPARTMENT is defined in the non-SQL defined demonstration database that can be installed with the system.

Definition: As defined at installation time:

- **The DEPARTMENT record** *owns* **the EMPLOYEE record**.

  In order to delete a department that owns employees, you must either disconnect or delete the department's employees. To delete a department along with its employees, you use a process command like:

  ```
  ERASE DEPARTMENT ALL MEMBERS.
  ```

- **The DEPARTMENT record is defined in the ORG-DEMO-REGION**.

Sample DEPARTMENT record layout: What is below shows the layout of the DEPARTMENT record as defined at installation time.

Note: The demonstration database and the DEPARTMENT record may be defined differently at your site.

```
Record:  DEPARTMENT             Version:  100

  Location mode:  CALC
  CALC field...:  DEPT-ID-0410
  Duplicates records are not allowed

     Element:  DEPT-ID-0410

             Picture:  9(4)
             Usage..:  Display

     Element:  DEPT-NAME-0410

             Picture:  X(45)
             Usage..:  Display

     Element:  DEPT-HEAD-ID-0410

             Picture:  9(4)
             Usage..:  Display
```

Index